MASTER EN SCIENCES INFORMATIQUES

Gestion de l'information d'un département informatique de production

Brutto, Calogero; Defreine, Pierre Jean

Award date:
2001

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX, NAMUR
INSTITUT D'INFORMATIQUE
RUE GRANDGAGNAGE, 21, B-5000 NAMUR (BELGIUM)

Gestion de l'information d'un département informatique de production

Calogero BRUTTO - Pierre Jean DEFREINE

Mémoire présenté en vue de l'obtention du grade de
Licencié en Informatique

Année académique 2000-2001

Résumé

Ce document traite du problème de la gestion de l'information au sein d'une communauté de personnes, à partir de l'étude d'un cas réel d'un département informatique de production.

Ce travail est composé de trois parties principales. Au cours de la première partie, nous avons étudié le contexte du problème et spécifié les besoins. Dans la deuxième partie, nous avons effectué une recherche des différentes solutions disponibles sur le marché et choisi celle qui convenait le mieux à notre problème. Dans la troisième partie, nous avons développé l'architecture technique de la solution qui a été retenue. Cette solution s'articule autour de XML (Extensible Markup Language) et J2EE (Java 2 Platform Enterprise Edition).

Nous concluons par les perspectives offertes par XML qui est devenu un standard tant pour la gestion de l'information que pour la communication au sens large.

Abstract

This paper deals with a community information management problem on concrete IT production department.

This work is split in three parts. The first part is about studying the problem context as well as specifying the needs. In the second part, we have searched some solutions, looking at the available market solutions and we have chosen one, which cope with our problem. In the third part, we have developed the technical architecture of the chosen solution that is based on XML and J2EE.

We conclude with the bright future that XML offer as a new standard for information management and communication.

Avant propos

Nous tenons à remercier,

Monsieur Ramaeckers, promoteur de ce mémoire, pour sa patience, sa disponibilité, ses conseils et ses encouragements.

Monsieur Vincent Collin, manager du département IOP chez Mobistar, pour nous avoir permis de réaliser ce mémoire dans les meilleures conditions ainsi que pour son soutien durant les différentes étapes de ce projet.

Tous les 'team leader' et 'team member' des différents services du département IOP, qui avec beaucoup de patience et de professionnalisme, nous ont consacré une partie de leur temps précieux pour nous permettre de cerner au mieux les différents problèmes auxquels ils sont confrontés.

Nos familles, car sans elles, il aurait été beaucoup plus difficile d'aller au terme de ces deux années d'études et de ce mémoire.

Les membres du Jury, pour l'intérêt qu'ils porteront à la lecture de ce mémoire.

Table des matières

Introduction

Les objectif de ce mémoire sont multiples.

Tout d'abord, l'objectif est de mettre en œuvre au travers d'un projet un maximum de méthodes et de techniques apprises tout au long de nos études, ce projet devant être réalisable chez notre employeur actuel: Mobistar.

Ensuite, il est impératif que le sujet du mémoire réponde à un besoin essentiel et réel du business. Ceci a pour but de confronter les contraintes habituelles rencontrées dans les entreprises (temps, argent...) aux théories apprises aux cours.

Enfin, le projet devra permettre une implémentation ultérieure qui nous permettra de valider sur du long terme les solutions que nous aurons apportées au problème. La facilité de maintenance et d'évolutivité de l'implémentation de la solution sera donc un point important.

1. Contexte

Le chapitre présente l'environnement du travail proposé, une description de la société Mobistar et les équipes concernées. Ce chapitre met également en évidence les motivations, les causes d'insatisfaction et les frontières du projet.

1.1. MOBISTAR

1.1.1. LA SOCIÉTÉ

Mobistar est le deuxième opérateur historique en téléphonie mobile. Depuis son lancement, Mobistar n'a cessé d'offrir des services de plus en plus étoffés autour de la téléphonie.

En ce qui concerne la téléphonie mobile, Mobistar a rapidement déployé une série de services tel que le voice mail, le GSM fax, la lecture de courrier électronique via GSM, les SMS et le WAP.

Le dernier de ces services, le GPRS (General Packet Radio Service), qui permet d'avoir un débit donné équivalent à une ligne téléphonique fixe sur un GSM de type GPRS, est déjà déployé sur toute la Belgique. Actuellement les tests d'intégration sont réalisés avec l'aide de clients désirant travailler rapidement avec cette technologie, l'offre commerciale GPRS pour tous les clients Mobistar devant être disponible dès la fin des ces tests.

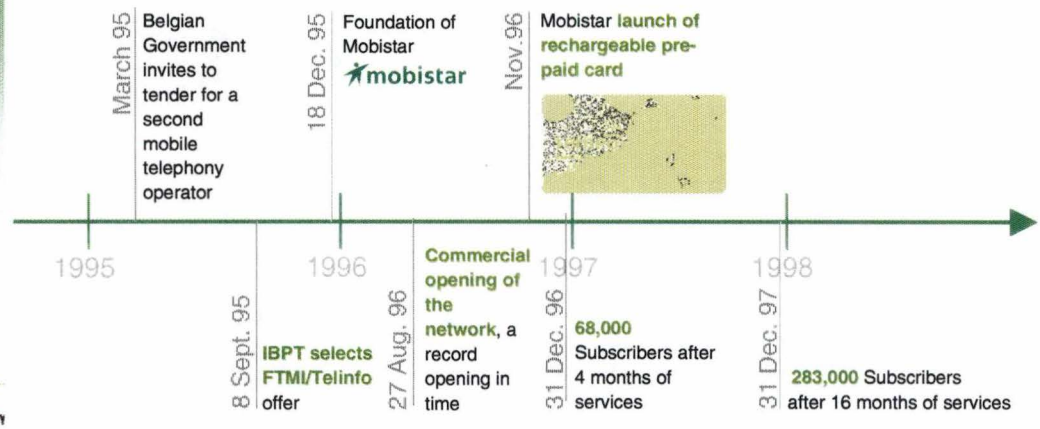
En ce qui concerne la téléphonie fixe, Mobistar a rapidement proposé des services tels le 1595.

Mobistar propose via sa filiale MCS (Mobistar Corporate solutions) une offre commerciale orientée Entreprise (Réseaux privés haut débit, accès à Internet,...).

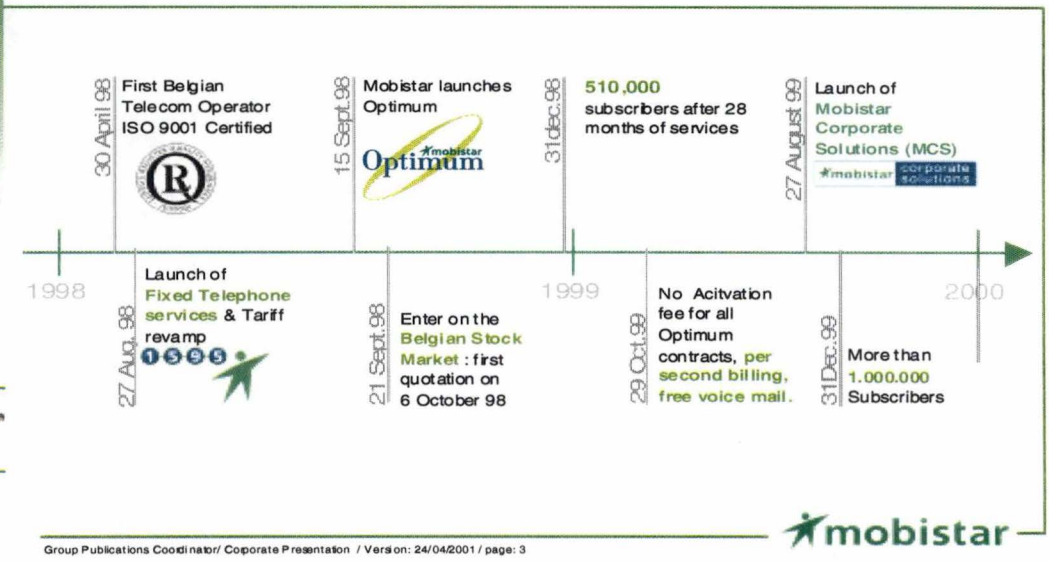
Enfin, Mobistar est un ISP (Internet service provider) à travers sa filiale Wanadoo.

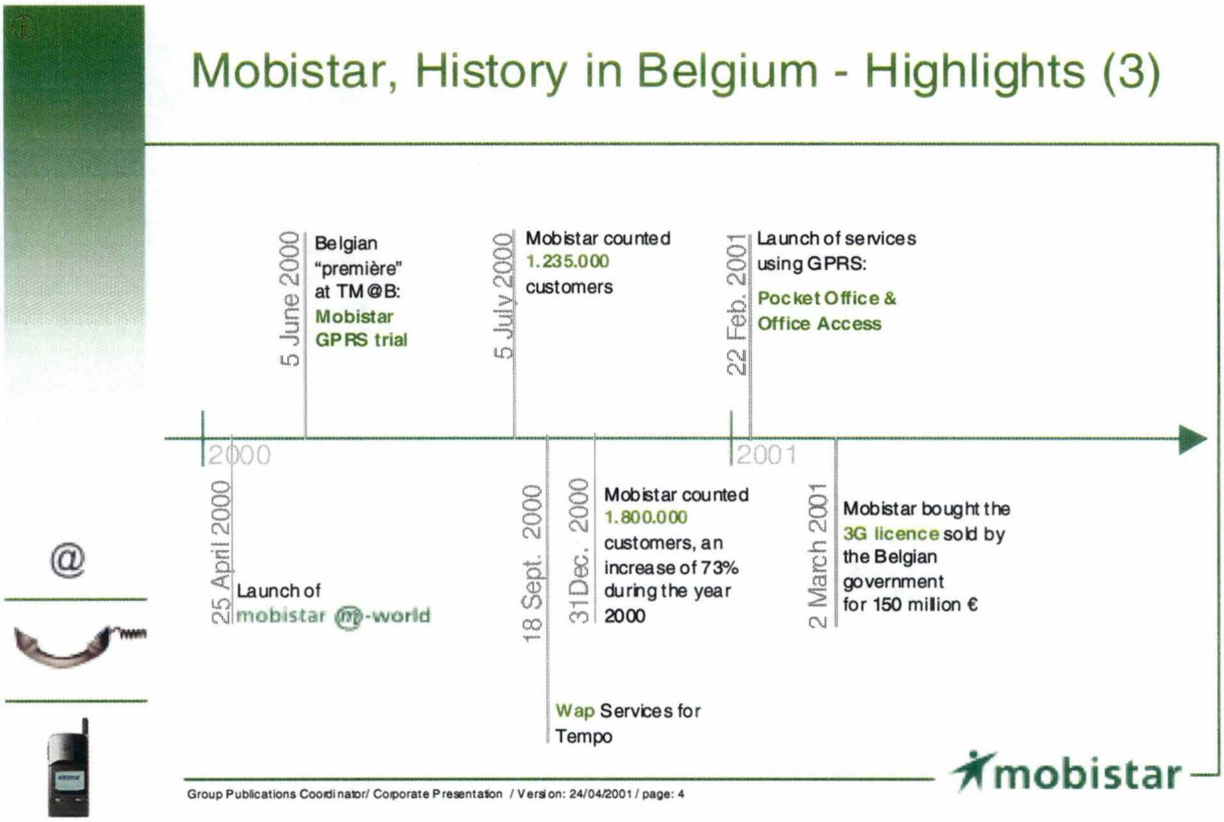
1.1.2. UN PEU D'HISTOIRE

Mobistar, History in Belgium - Highlights (1)

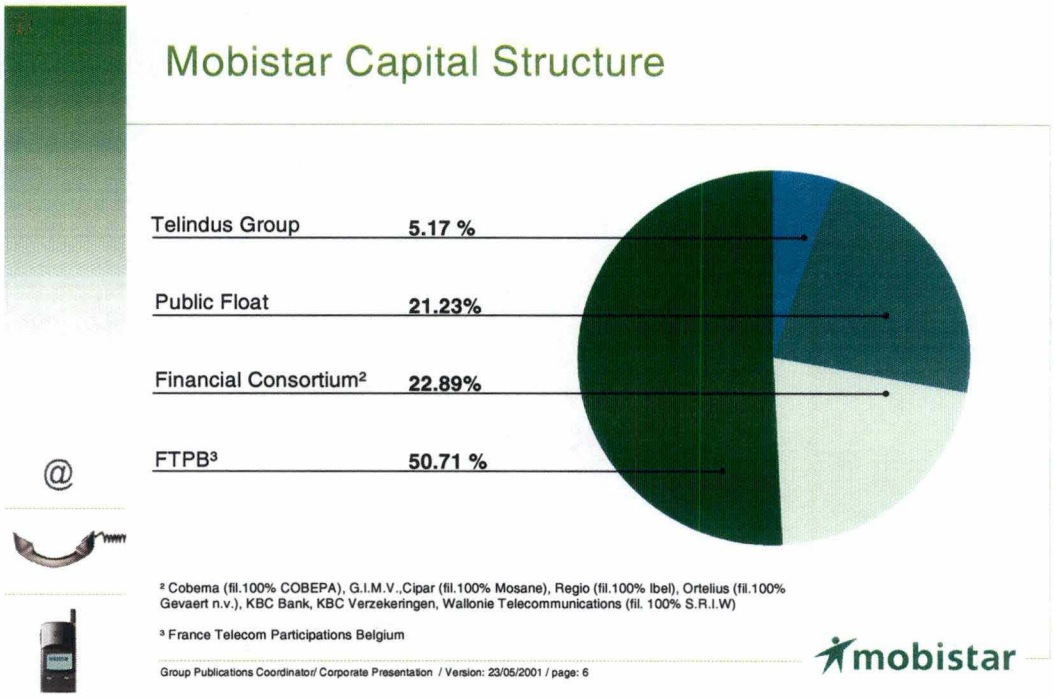


Mobistar, History in Belgium - Highlights (2)




Mobistar, History in Belgium - Highlights (3)
2000
25 April 2000
Launch of
mobistar m-world
5 June 2000
Belgian
"première"
at TM@B:
Mobistar
GPRS trial
5 July 2000
Mobistar counted
1.235.000
customers
18 Sept. 2000
Wap Services for
Tempo
31Dec. 2000
Mobistar counted
1.800.000
customers, an
increase of 73%
during the year
2000
2001
22 Feb. 2001
Launch of services
using GPRS:
Pocket Office &
Office Access
2 March 2001
Mobistar bought the
3G licence sold by
the Belgian
government
for 150 million €
mobistar
Group Publications Coordinator/ Corporate Presentation / Version: 24/04/2001 / page: 4

1.1.3. LES ACTIONNAIRES

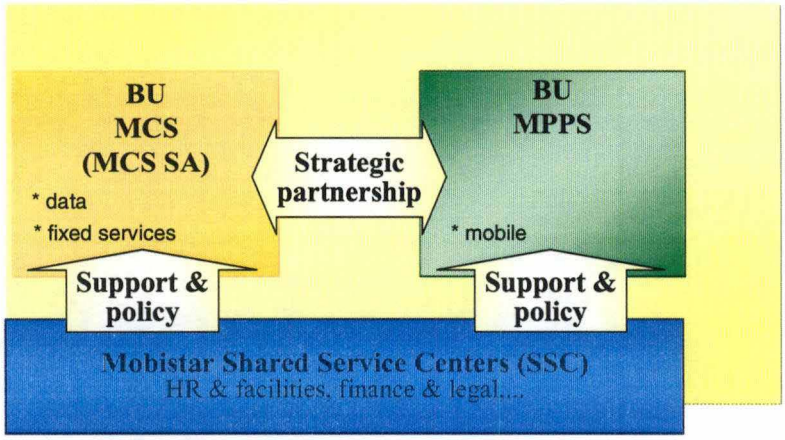


1.1.4. STRUCTURE DU GROUPE MOBISTAR S.A.

Le groupe Mobistar S.A. est composé de trois Business Units :

- M.C.S. (Mobistar Corporate Solution) s'occupe de la commercialisation des solutions de téléphonie fixe et du transport de données.
- M.P.P.S. (Mobistar Personal and Professional Solutions) s'occupe de la téléphonie mobile.
- M.S.S.C. (Mobistar Shared Service Centers) procure aux différentes Business Units du groupe un ensemble de services (Finance, Ressource Humaine, Informatique, ...); ces services sont appelé 'shared services'.
- M.R.S (Mobistar Residential solutions) s'occupe de la commercialisation des services résidentiels.

The Mobistar Structure

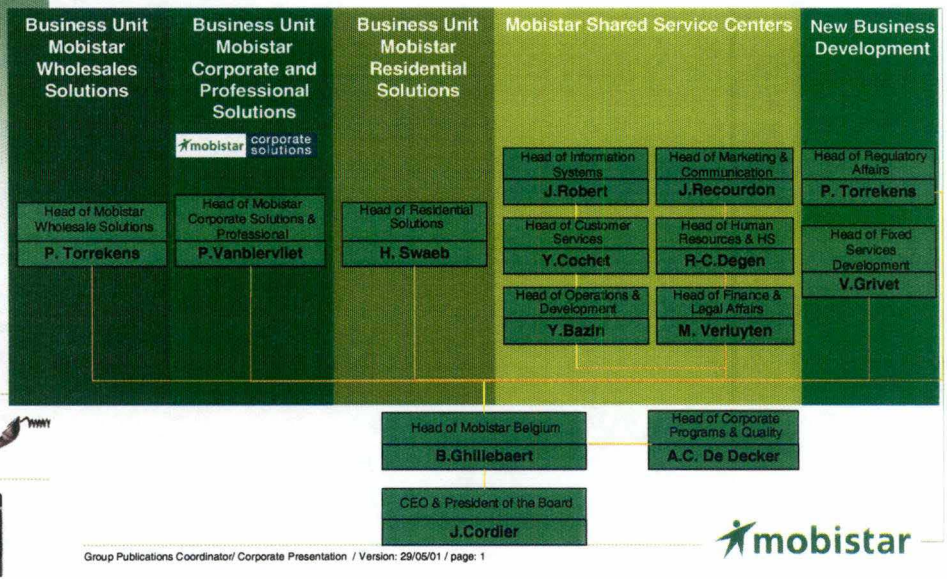


Group Publications Coordinator/ Corporate Presentation / Version: 29/05/01 / page: 1

mobistar

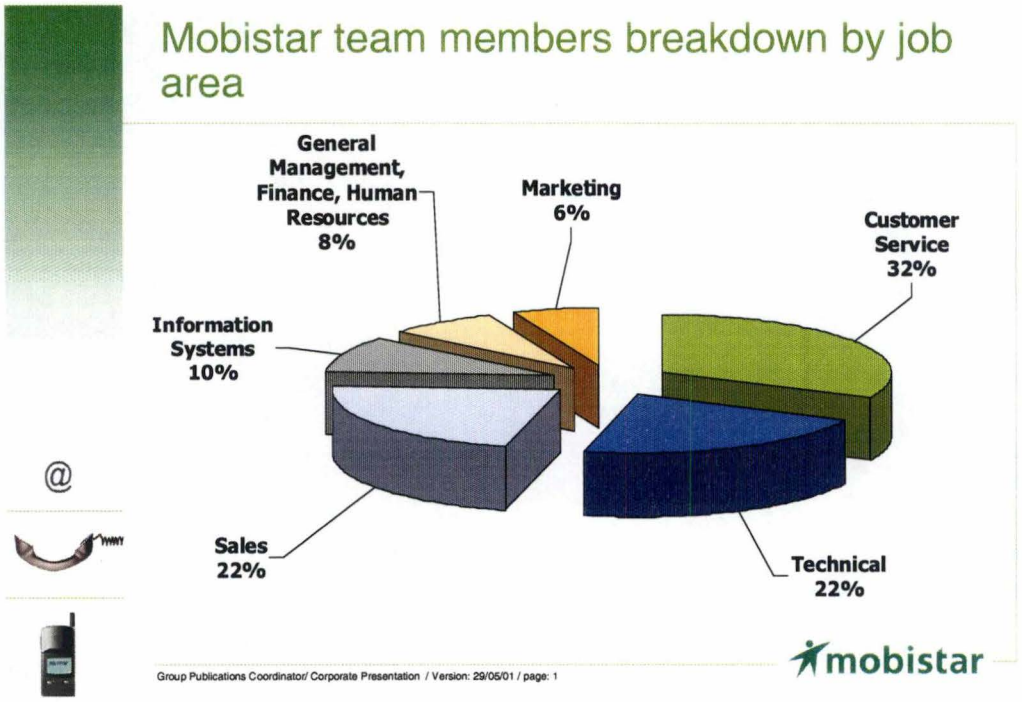
mobistar I know why.

"One face to the Customer"



Group Publications Coordinator/ Corporate Presentation / Version: 29/05/01 / page: 1

mobistar



Le département IOP (informatique opérationnelle) fait partie du service Informations Systems.

1.1.5. LE DÉPARTEMENT IOP

Le S.I. (système d'information) qui fait l'objet de cette étude est destiné essentiellement au département IOP (informatique opérationnelle).

Ce département s'occupe de ce que l'on appelle communément la "Production". IOP n'est responsable d'aucun développement informatique. Ce département a pour tâche de s'assurer du bon fonctionnement des chaînes d'applications, des architectures applications, des systèmes informatiques (serveurs), des bases de données ainsi que de la sécurité informatique.

Ce département fournit ses services aux autres départements de Mobistar:

- IOP gère les systèmes ainsi que les bases de données du département DOD (Deployment and Operational Department).
- IOP s'assure du fonctionnement des applications reçues du département ISD (information services department).
- IOP est responsable de la mise en place et du suivi de l'infrastructure e-business.
- IOP est responsable de la sécurité informatique au sein de Mobistar.

1.1.6. LES DIFFÉRENTES ÉQUIPES CONCERNÉES

Le département IOP est constitué de plusieurs équipes. Une équipe est constituée approximativement de dix personnes.

Voici les équipes constitutives du département IOP:

Management

Le management du département IOP est responsable des grandes orientations, du bon fonctionnement ainsi que de l'efficacité du département et donc des différentes équipes qui le constituent. Cette équipe est constituée de tous les team-leaders des autres équipes.

Methode et organisation (MET)

L'équipe MET définit les méthodes de travail pour le département. Cela implique la définition de processus, de méthodes, de templates, la gestion des documentations applications, la gestion des dashboards

Supervision (IOT)

L'équipe IOT est responsable de la supervision des chaînes applications. A partir d'une console centrale, cette équipe planifie,

synchronise et vérifie la bonne exécution des applications. Cela concerne aussi bien les applications OLTP (interactives) que les applications batch. En cas de problème de quelque nature que ce soit (application, système, réseau, électrique...), des alertes sont remontées à la console centrale et, en fonction du type du message, l'opérateur se charge de contacter l'équipe responsable de résoudre cette panne.

Sécurité (SCY)

L'équipe SCY est responsable de la définition des 'Security Policy' de Mobistar.

Cela concerne:

- L'accès logique aux systèmes d'information (gestion des accès aux applications par les utilisateurs)
- La définition des règles d'accès aux systèmes et aux différents réseaux (firewall, authentification, LDAP, remote access...) par les employés de Mobistar.
- La définition des règles de connexion informatique entre Mobistar et certains de ses fournisseurs ou clients.

Les équipes applications (APP)

- ***Rating and billing (APP/RAB)***

L'équipe RAP est responsable de l'exécution correcte des applications concernant les domaines du Rating et du Billing ainsi que la valorisation des communications téléphoniques (transformer les compte-rendus des appels de nos clients en factures).

- ***APP/SUB***

L'équipe SUB est responsable de l'exécution correcte des applications liées à la gestion de notre clientèle. Cela concerne par exemple les applications liées au call-center (hot line GSM Mobistar) ainsi que les campagnes marketing d'appel de nos clients pour leur proposer de nouveaux services.

- ***Data Warehouse and internal system (APP/DIS)***

L'équipe DIS est responsable de l'exécution correcte de l'application Data Warehouse ainsi que des applications interactives non orientées clientèle.

Database (DBA)

L'équipe DBA est responsable du bon fonctionnement de toute base de données chez Mobistar (Oracle, SqlServer...). Cela implique entre autre un suivi journalier de l'espace pris par les bases sur chaque système, une vérification du bon déroulement des sauvegardes de la nuit... Cette équipe gère près de 200 bases de données dont la taille varie en moyenne de 40 à 200 gigabytes et dont la plus importante fait1.4 terabytes.

Système (SYG)

L'équipe SYG est responsable du bon fonctionnement des serveurs, des sauvegardes de toutes les données des serveurs, des espaces disque, de l'outil Unicenter ainsi que de l'infrastructure e-mobistar. Les serveurs sont au nombre de 100. Ce sont essentiellement des serveurs UNIX (SUN et Compaq) et depuis peu des serveurs NT qui vont de l'entrée de gamme jusqu'au plus puissants de chaque marque. L'infrastructure e-mobistar est constituée d'équipement permettant un grand nombre de connexions sécurisées à nos sites tant Web que Wap

(load balancer, webcache, firewall, web server, application server, proxy...).

1.2. LE PROJET

Ce projet concerne la consolidation et la mise à disposition des informations nécessaires aux équipes du département IOP.

Ces informations sont nécessaires aux équipes pour :

- La mise en production des versions et systèmes qui composent le SI
- La maintenance des composants du SI
- La rédaction et suivi des E.R.
- La définition de standards, gestion de projets
- Le travail journalier sur les différentes chaînes de productions

1.2.1. LA MOTIVATION DU PROJET

Du fait de la croissance de Mobistar, le nombre d'équipes intervenant dans la réalisation du travail quotidien, comme dans le suivi des nouveaux projets est de plus en plus important.

De plus, la mise en place et la maintenance journalière des applications nécessitent de plus en plus de compétences diverses, donc de plus en plus d'équipes doivent intervenir et donc coopérer pour mettre en place et pour maintenir les solutions informatiques nécessaires au bon déroulement des activités de Mobistar.

Cela signifie que les problèmes de communications et d'échanges d'informations entre les différents intervenants deviennent de plus en plus complexes et de plus en plus lourds à gérer.

Il a donc été décidé d'envisager de mettre en place des outils de gestion de ces informations. Les départements informatiques mettent en

général à disposition des autres départements de l'entreprise des systèmes d'information de plus en plus sophistiqués alors que le département informatique lui même gère généralement assez mal son information. Il n'existe d'ailleurs pas sur le marché d'outils intégrés permettant cette gestion alors qu'une multitude d'outils existe pour gérer toutes les facettes des besoins habituels des entreprises (comptabilité, facturation, clientèle...).

La motivation du projet est de mettre en place un S.I. capable:

- De mettre en place rapidement un outil permettant à tout Team member de mettre à disposition de l'information structurée pour son équipe ainsi que pour d'autres équipes.
- De pouvoir manipuler l'information de manière intuitive.
- De définir aisément des "templates" pour structurer et incorporer l'information gérée par toutes les équipes.
- De gérer l'information appartenant à toutes les équipes à un seul endroit.
- D'intégrer aisément, de manière manuelle ou automatique, toutes nouvelles sources d'information, la structure de l'information devant rester souple et malléable.
- D'éviter un encodage inutile quand l'information peut être récoltée de manière automatique Par exemple: la liste et le pourcentage d'espace libre des filesystems de tous les systèmes UNIX.
- De gérer les liens entre l'information des différentes équipes.

La motivation du projet **n'est pas** :

- D'analyser l'ensemble des flux. Cette tâche serait trop ardue et serait obsolète dès qu'elle serait terminée.
- D'analyser l'ensemble des structures d'information. Cette tâche serait également obsolète dès qu'elle serait terminée.

1.2.2. CAUSES D'INSATISFACTION

L'absence de structure et le stockage éparpillé de l'information pose différents problèmes:

- L'information existe mais est inaccessible
- Nous ne sommes jamais sûrs qu'elle est à jour
- Nous ne savons pas que l'information existe
- Des systèmes de gestion de la documentation hétérogène ont été mis en place par les différentes équipes.

Même si des initiatives ont déjà été prises pour mettre en place des procédures communes de gestion de l'information (partage de disques réseaux, standard de document).

Les solutions mises en place par les différentes équipes sont propres aux équipes et développées sans concertation et sans standard. Outils différents, access, WEB, word, il est impossible d'avoir une vue globale et consolidée de l'information gérée par toutes les équipes.

- Il est impossible de rechercher de l'information sans intervention humaine.
- Il est impossible de lier l'information intra-équipe ainsi que inter-équipe.
- Cela rend impossible toute exploitation rapide, voir automatique, de l'information.
- Cela ne permet pas de se rendre compte du volume total d'informations réellement gérées.

1.2.3. DÉFICIENCES

- Manque de publicité.
- Manque de communication entre équipes.
- Manque de sécurité.
- Manque de qualité.

- Manque de standards.
- Information difficilement accessible.

1.2.4. LA FRONTIÈRE DU PROJET

- Le projet ne concerne que le département IOP.
- La mise en place du projet doit se faire avant la fin 2001.
- Le projet doit tenir compte des objectifs corporate, par exemple une volonté de généraliser le bureau sans papier par l'utilisation de produit comme documentum.

2. Etude critique du S.I. existant

2.1. LE S.I. EXISTANT

Actuellement les équipes gèrent des documents sous les formes et supports les plus divers. Papier, électronique, structuré, non structuré, dans des fardes, en ligne via le WEB ou des disques partagés.

Devant l'accroissement du volume de documents reçus, générés, échangés, chaque équipe a commencé à développer sa propre solution. Il existe donc différents standards, outils qui sont utilisés au sein de chaque équipe.

Par exemple :

- L'équipe SYG utilise un WEB site pour centraliser l'information sur les standards et les set-up des différents systèmes gérés par cette équipe. Le problème ici, est que ces données sont statiques et que le site a été développé en utilisant des particularités du logiciel FrontPage.

- L'équipe DBA utilise un disque partagé par l'équipe. Elle partage également le même site WEB que l'équipe SYG. Les documents de standards sont stockés sur le disque partagé. Le site web contient des rapports sur les bases de données ainsi que les résultats des différents jobs journaliers. Les données du site WEB DBA sont entièrement dynamiques. Elles sont générées en utilisant des "cgi" qui renvoient des pages HTML. Les rapports sont exécutés via un "scheduler" et poussés sur le site WEB.

- Dans d'autres équipes, un autre site WEB est utilisé et les rapports sont sous la forme de rapports excel.

- Il existe aussi des équipes qui utilisent un mixte entre des données papiers et des données électroniques, une secrétaire étant responsable de la centralisation des données papiers.

En ce qui concerne le flux d'informations entre IOP et les autres départements, des outils ont été mis en place pour répondre à des problèmes spécifiques. Par exemple, le suivi des 'Evolutions Request' entre les équipes de développement et les équipes applications chez IOP.

La gestion des ces flux ne fait pas partie du SI qui doit être analysé.

Comme le montre les différents exemples décrits ci-dessus, le S.I. existant de gestion de l'information au sein d'IOP n'est ni entièrement informatique ni même entièrement formalisé. Il ne s'appuie que sur la bonne volonté des membres de chaque équipe et sur la constatations des différents Team Leader qu'un minimum de structure devait être mis en place.

Ce manque de structuration s'explique par le fait que Mobistar est une jeune compagnie en perpétuelle restructuration. Cela est dû en grande partie à son accroissement et à la diversification des produits proposés aux clients. En effet l'architecture informatique pour gérer 500.000 clients n'a rien à voir avec celle nécessaire pour gérer 1.000.000 de clients, qui n'a elle-même rien à voir avec celle nécessaire pour gérer 2.000.000 de clients. De plus les compétences nécessaires pour faire de la téléphonie Mobile, n'ont rien avoir avec la complexité introduite par les VPN, le e-business, GPRS, ...

Le point sur lequel s'est focalisé Mobistar ces dernières années était de mettre en place l'architecture nécessaire pour supporter cet accroissement de clientèle et l'introduction de ses multiples nouveaux services. Mobistar en est arrivé maintenant à un point où l'infrastructure est réalisée, mais pour l'utiliser au mieux de ses possibilités, il faut absolument améliorer la disponibilité de l'information dans les équipes et entre les équipes.

Cette meilleure disponibilité de l'information permettra d'améliorer la vue globale et/ou détaillée des différents systèmes composants le SI.

Cette vue globale est nécessaire pour le management, mais aussi pour les différents intervenants dans le but de mesurer de la façon la plus rapide et la plus correcte l'impact de telle ou telle modification. Par exemple, quelles chaînes applications seront impactées si je déménage une des salles informatiques ; ...

La vue détaillée est également nécessaire pour tous les intervenants de IOP. Pour gérer au mieux les impacts des différentes actions qu'ils sont amenés à réaliser dans le cadre de leur travail quotidien. Par exemple, quelle application est impactée si une base de donnée doit être momentanément arrêtée, qui doit être prévenu dans le mail d'incident.

2.2. ETAT DES LIEUX

2.2.1. L'INFORMATION EST STOCKÉE SOUS DIVERS FORMATS ÉLECTRONIQUES.

- fichiers texte
- fichiers excel
- fichiers word
- fichiers powerpoint
- fichiers HTML
- ...

2.2.2. L'INFORMATION EST STOCKÉE AVEC DIVERSES MÉTHODES DE STOCKAGE

- Disque local à la station de travail de l'utilisateur
- Disques réseaux partagés et localisés sur différents serveurs
- Server web
- Mailbox partagée par différents utilisateurs
- Papier
- ...

2.2.3. L'INFORMATION EST ESSENTIELLEMENT ÉCHANGÉE OU PARTAGÉE VIA

- Mail électronique.
- Disques partagés.
- Mail interne.

La pluspart de ces informations avec lesquelles les différentes équipes sont confrontées, sont des informations que l'on peut qualifier de structurées. Un document structuré est un document dont on peut qualifier l'ensemble des éléments qui le constitue.

Exemple de document structuré disponible chez Mobistar :

- des bases de données,
- des systèmes,
- des liens entre les applications, les bases et les systèmes,
- des modes opératoires,
- des manuels d'installation,
- des manuels de gestion d'erreur,
- des manuels d'enchaînement de tâches,
- ...

Les informations structurées, les liens entre informations et la mise à disposition sont **le sujet du présent travail.**

3. Définition du projet

Ce chapitre présente les objectifs sous les angles organisationnels, informationnels, et qualitatifs. Les activités concernées sont ici décrites ainsi qu'un schéma général de l'outil souhaité. Enfin les contraintes fonctionnelles sont spécifiées.

3.1. LES OBJECTIFS DE L'ORGANISATION

Les objectifs organisationnels sont :

- De mettre en place rapidement un outil permettant à tout team member de mettre à disposition de l'information structurée et de pouvoir la manipuler de manière intuitive.
- De définir aisément des "templates" pour structurer et incorporer l'information gérée par toutes les équipes; ainsi l'information se trouvera structurée et à un seul endroit.
- D'intégrer aisément, de manière manuelle ou automatique toutes nouvelles sources d'information, la structure de l'information devant rester souple et malléable.
- D'éviter un encodage inutile quand l'information peut être récoltée de manière automatique.
- De gérer les liens entre l'information des différentes équipes.
- De consolider l'information des différentes équipes.
- De permettre une vue globale et détaillée de l'information gérée par les différentes équipes.
- De mettre à disposition aisée à toutes les équipes du département de l'information structurée.
- De réduire les coûts de fonctionnement par un accès aisé à l'information.
- D'améliorer le service rendu à nos clients internes et externes.

3.2. LES OBJECTIFS INFORMATIONNELS

3.2.1. STRUCTURE DE L'INFORMATION

- La structure des informations stockées par le S.I. doit pouvoir évoluer à tout moment sans requérir une évolution du S.I. La structure de l'information doit donc être paramétrable par

l'utilisateur par l'utilisation de ce que nous appellerons un "template".

- Un template contient la description de la structure d'un document ainsi que la définition des éléments qui composent la structure.
- Le S.I. est principalement composé de documents structurés que l'on appellera instance.
- Une instance d'un template est un document conforme à la structure du template dont les éléments ont été remplacés par des données.
- Les éléments doivent être typés et identifiables.
- Un élément peut être lui-même composé de sous éléments.
- Des liens entre instance de différentes sémantiques (dépendant de, concerne, documente,...) doivent pouvoir être associés aux instances.
- Pour intégrer les demandes fonctionnelles de gestion de liens inter documents et de possibilités de commentaires, il est prévu que les templates allient:
  - une partie fixe, la gestion des liens 'vers/de',
  - une partie commentaire sur le template,
  - une ou plusieurs parties métier dont les éléments sont plus ou moins spécialisés.

    Cela peut simplement consister en une définition simple de paragraphe de document, ou en une structure très précise, qui définit par exemple pour l'équipe base de données opérationnelles la structure physique d'un base donnée, autrement dit les différents fichiers qui la compose (tablespaces + fichier, initora, control file) ainsi que les différents paramètres contenus dans les fichiers de configuration (taille de block, utl_file_dir, ...) ou simplement leur localisation sur un système de fichier.

- La mise en page d'une instance d'un template ne fait pas partie de sa définition. Il doit être possible d'appliquer à une instance d'un template une multitude de mise en page différentes.

3.2.2. QUALITÉ DE L'INFORMATION

- La qualité de l'information ne dépend pas du S.I. Elle dépend uniquement de l'utilisateur responsable de cette information. Le S.I. n'est responsable que du stockage et de la restitution de l'information. Il n'est pas responsable ni du contenu, ni de sa structure.
- De ce fait, l'information se doit d'être attachée à un utilisateur ou à une équipe qui sera responsable de sa crédibilité.
- C'est de la responsabilisation que dépendront la qualité, l'âge, la précision, la pertinence et la crédibilité de l'information.
- En fonction de la plus ou moins grande spécialisation des éléments qui constituent une instance, et dans le but d'éviter les encodages manuels de données, il faut envisager de récupérer, de générer ou de rapatrier automatiquement les données pour lesquelles cela est possible.
- Les bases de données ou les systèmes sont de bons candidats pour une automatisation complète de la découverte des éléments qui les composent.
    - En effet pour une base de données, toutes les informations sont normalement accessibles via le dictionnaire des données.
    - Pour un système Unix, toutes les informations sur sa configuration, sont disponibles soit via des rapports fournis par les fournisseurs, soit par l'utilisation d'un ensemble de commandes Unix qui permettent de récupérer toutes les informations.

3.2.3. CARACTÉRISTIQUES DES PROCESSEURS

Utilisabilité

- Le S.I. doit être utilisable par tout utilisateur sans requérir de formations.
- La visualisation de l'information ainsi que sa mise à jour doit se faire via la même interface utilisateur.
- L'utilisateur doit pouvoir passer simplement de la visualisation de l'information à sa mise à jour.

Performance

- Le S.I. doit répondre à toute requête (visualisation, update...) en moins de 5 secondes.

Disponibilité

- Le S.I. doit avoir un taux de disponibilité de 99% en mise à jour. Cette moyenne sur un an est la moyenne standard observée et requise chez Mobistar.
- Le S.I. doit avoir un taux de disponibilité de 100% en lecture car ce S.I. contiendra toute l'information du DRP. Un dédoublement du S.I. sur les deux sites de production de Mobistar est à prévoir.

Sécurité

- L'accès aux informations doit être contrôlé par une authentification de l'utilisateur.
- L'acquisition automatique d'information sur des S.I. externes doit se faire de manière sécurisée (authentification, chiffrement des communications, stockage chiffré des informations).
- Les principales informations de sécurité au niveau d'une instance d'un template sont :

- Qui peut référencer une instance d'un template ?
- Qui peut lire une instance d'un template ? Par exemple: l'équipe DBA pourrait ne pas pouvoir créer un lien dans sa documentation référencant les équipement réseaux mais pourrait avoir accès en lecture à toutes les informations concernant ces mêmes équipement réseaux. En effet, les bases de données reposent sur le bon fonctionnement de serveurs qui eux mêmes reposent sur le bon fonctionnement d'équipements réseaux. Les bases de données peuvent donc référencer des serveurs mais pas des équipement réseaux.
- Qui peut modifier une instance d'un template ?

- Dans le cas qui nous concerne, il faut en plus gérer des notions de sécurité du contenu d'une instance. Comme une instance de template est une structure sous forme d'arborescence, il est possible de gérer des attributs de sécurité sur chaque élément ! Si l'on veut gérer la sécurité au niveau de l'instance, il suffit de mettre la sécurité sur l'élément racine.

3.3. ACTIVITÉ CONCERNÉE

- Les différentes équipes du département d'IOP ont toutes une tâche commune qui est la documentation des éléments gérés par chaque équipe.

3.4. LES CRITÈRES D'EFFICACITÉ

3.4.1. CRITÈRES D'EFFICACITÉ ORGANISATIONNELLE

- L'accès aux informations privées de l'équipe de l'utilisateur permettra d'améliorer la communication intra-équipe. La plupart des équipes gérant plusieurs applications distinctes travaillant dans une

même équipe pourront ainsi partager l'information à travers un seul et même canal.

- L'accès aux informations publiques de toutes les équipes du département permettra d'améliorer la communication inter-équipes. Les différentes équipes de notre département ayant très régulièrement besoin d'information provenant des autres équipes, le S.I. leur permettra d'accéder plus aisément et plus complètement à l'information des autres équipes.
- Le S.I. permettra au management d'avoir une vue complète, précise et à jour des informations globales et détaillées de chaque équipe.
- Les informations doivent être accessibles via un moteur de recherche.

3.4.2. CRITÈRES D'EFFICACITÉ ÉCONOMIQUE

Il est difficile de calculer l'impact économique direct du nouveau S.I. Il est certain qu'un gain de productivité substantiel est à prévoir. Ainsi qu'une meilleure vue du management sur les différentes équipes du département ce qui aura pour conséquence d'aider les directeurs et les managers lors des choix stratégiques.

3.4.3. CRITÈRES D'EFFICACITÉ DE RÉALISATION

Durée de vie

La durée de vie du nouveau S.I. peut-être estimée à 4 ans au minimum, étant donné le caractère non figé des structures des informations stockées ainsi que l'utilisation de technologies standard de grande diffusion.

Délai de mise en exploitation

La mise en exploitation du nouveau S.I. doit se faire avant la fin de l'année 2001.

Scénarios de mise en exploitation

Un scénario de mise en exploitation détaillé est inutile ici car il n'y aura pas de S.I. existant à reprendre. Les informations actuellement exploitées par chaque équipe devront être intégrées manuellement et sous la responsabilité de chaque équipe dans le nouveau S.I.

Il est effectivement hors du scope de ce projet de contraindre toute personne ou toute équipe à mettre une partie ou toutes ses informations dans le S.I. Cela doit se faire sur base volontaire ou par une pression inter-équipe.

Le nouveau S.I. doit être perçu comme un outil à la disposition de toute équipe ou tout team-member et non pas comme un outil contraignant. Il va de soit que chaque team-leader sera responsable de la qualité et de la quantité d'informations que son équipe incorporera dans le S.I.

L'outil doit être assez évolutif que pour permettre la création de structures d'informations à la demande. De ce fait, la mise en exploitation se fera au fil de l'eau, au gré des besoins. L'analyse permettra de retrouver les équipes qui seront les plus intéressées par l'utilisation de cette application. Ces équipes seront les premières à intégrer l'application. A terme, toutes les équipes devraient y trouver leur place.

3.5. EXIGENCES FONCTIONNELLES

Les besoins fonctionnels résumés ici font suite aux auditions individuelles de tous les team leaders de toutes les équipes appartenant à IOP ainsi que toutes les personnes étant le plus impliquées dans la gestion actuelle (manuelle ou non) de l'information de l'une ou l'autre équipe.

Les besoins fonctionnels peuvent se grouper en trois catégories:

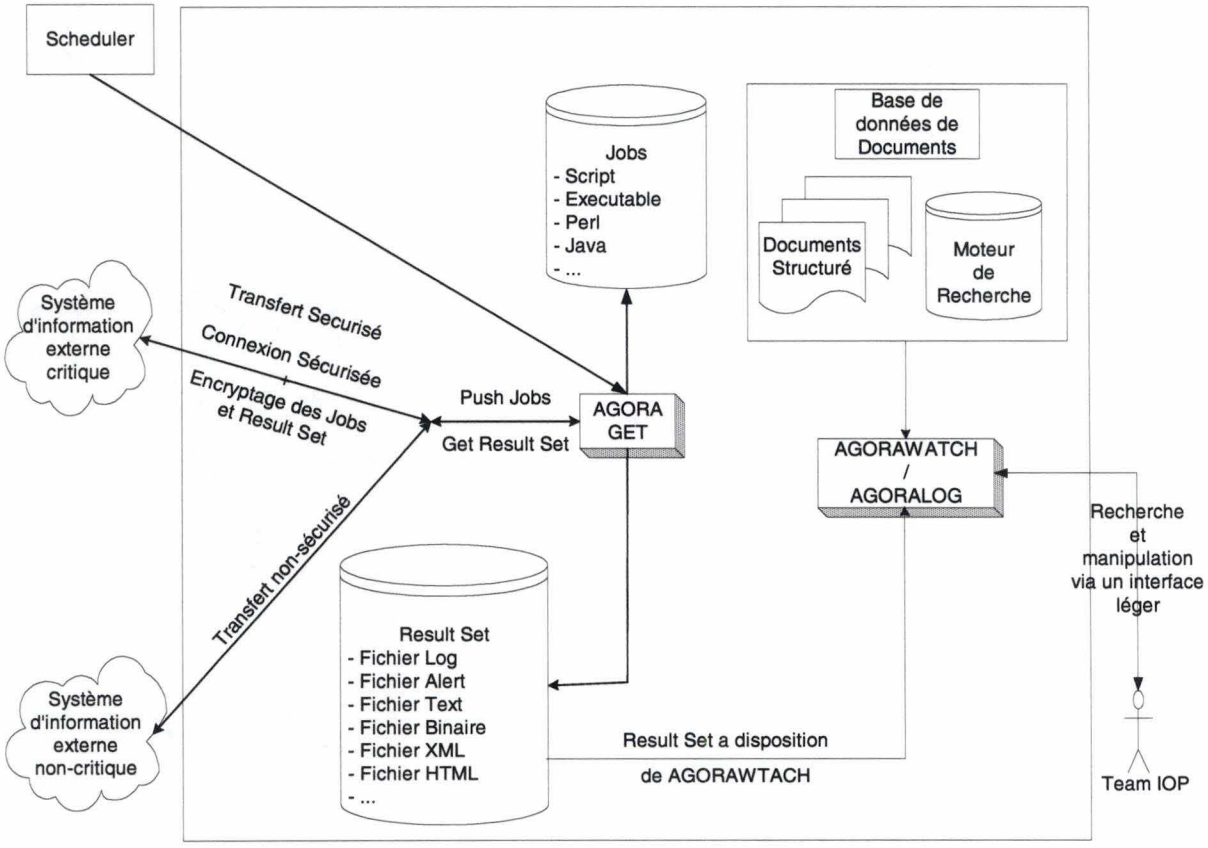
- Les besoins liés à la récolte automatique de l'information que nous regrouperons sous l'appellation : AGORAGET.
- Les besoins liés à l'accès et à la mise à jour manuelle de l'information que nous regrouperons sous l'appellation : AGORAWATCH.
- Les besoins liées au stockage de l'information des activités journalières des différentes équipes que nous regrouperons sous l'appellation : AGORALOG.

Ces catégories ont pour but de regrouper ensemble les spécifications fonctionnelles se rapportant à un même type de besoin ainsi que de leur affecter un ordre de priorité pour leur mise en place :

	Priorité de mise en place
AGORAWATCH	1
AGORALOG	2
AGORAGET	3

Table 3-1

3.5.1. SCHÉMA GÉNÉRAL DU S.I.



3.5.2. CONTRAINTES FONCTIONNELLES / NON FONCTIONNELLES AGORAWATCH

Identifiant	Contraintes fonctionnelles	Catégorie
AWAT F1	F1.1 Le S.I. doit permettre à un utilisateur de visualiser ainsi que de modifier l'information structurée contenue dans le S.I. F1.2 Le S.I. doit permettre de rechercher l'information stockée dans le S.I.	Visible
AWAT F2	F2.3 Le S.I. doit permettre de définir des templates pour structurer l'information. F2.3 Un template regroupera les informations ayant le même sujet.	Visible
AWAT F3	Une instance doit être un document conforme à la structure du template dont les éléments ont été remplacés par des données.	Visible
AWAT F4	La visualisation d'une instance doit être soumise à un contrôle d'accès par groupe d'utilisateur.	Visible
AWAT F5	La manipulation d'une instance doit être soumise à un contrôle d'accès par groupe d'utilisateur.	Visible
AWAT F6	Chaque employé sera défini comme appartenant à une "équipe".	Visible
AWAT F7	F7.1 Une instance doit pouvoir contenir des niveaux de titres F7.2 Une instance doit pouvoir contenir des textes F7.3 Une instance doit pouvoir contenir des fichiers F7.4 Une instance doit pouvoir contenir des hyperliens F7.5 Une instance doit pouvoir contenir des liens vers d'autres instances F7.6 Une instance doit pouvoir être typée par des attributs dont la liste des valeurs est préfixée. F7.7 Une instance doit pouvoir contenir des images. F7.8 Une instance aura une équipe qui sera son propriétaire. Cette équipe sera la responsable du contenu. F7.9 Une instance est composée d'éléments qui pourront chacun avoir des attributs de sécurité (droits d'accès en lecture et en écriture)	Visible
AWAT F8	F8.1 Une instance doit pouvoir être annotée par un utilisateur appartenant à une équipe qui peut visualiser le document F8.2 Une instance doit pouvoir contenir des annotations même si l'annotateur ne possède pas de droits de modification de l'instance F8.3 Une annotation sera toujours visible par le	Visible

	propriétaire du document F8.4 Une annotation sera toujours visible par l'annotateur F8.5 Une annotation pourra être visible par d'autres équipes que celui du propriétaire du document si l'annotateur le décide.	
AWAT F9	F9.1 Certaines informations doivent être stockées cryptéés. F9.2 Ces informations doivent également transiter jusqu'à l'utilisateur de manière cryptée.	Caché
AWAT F10	Le S.I. doit pouvoir permettre de définir des liens bidirectionnels entre instances d'entités.	Visible
AWAT F11	Toute manipulation de l'information des instances sera logée pour permettre un suivi des actions réalisées.	Caché
AWAT F12	Le S.I. doit permettre de copier/coller des informations à partir de l'interface graphique de l'utilisateur à destination du S.I.	Caché
AWAT F13	Le S.I. doit permettre la création de graphiques à partir de données stockées dans des fichiers plats.	Caché
AWAT F14	Le S.I. doit être dupliqué sur un deuxième serveur localisé sur un site différent (l'un sera à Charleroi et l'autre à Bruxelles). L'un sera MASTER et l'autre SLAVE.	Caché
AWAT F15	L'information entre les deux sites (AWAT F14) sera synchronisée une fois par jour.	Caché
Identifiant	Contraintes non fonctionnelles	
AWAT NF 1	Les droits d'accès doivent être stockés dans le LDAP de Mobistar.	
AWAT NF 2	Le S.I. doit être accessible par "k-village" et avoir le même "look & feel"	
AWAT NF 3	Les informations actuelles des équipes doivent être migrées (manuellement ou automatiquement) vers le nouveau S.I.	
AWAT NF 4	NF 4.1 L'interface utilisateur de l'application sera un browser web (internet explorer ou netscape). NF 4.2 Aucune application (autre que le browser) ne devra être installée sur le poste de l'utilisateur.	

Table 3-2

3.5.3. CONTRAINTES FONCTIONNELLES / NON FONCTIONNELLES LOG BOOK

Identifiant	Contraintes fonctionnelles	Catégorie
LB F1	Le S.I. doit permettre d'encoder des informations au sujet des incidents journaliers que l'utilisateur a dû traiter.	Visible
LB F2	Le S.I. doit permettre d'encoder des informations au sujet des incidents survenus lors des gardes (soirée et week-end) que l'utilisateur a effectuées.	Visible
LB F3	Le S.I. doit permettre d'encoder des informations au sujet des demandes de travail que l'utilisateur a dû traiter.	Visible
LB F4	Les incidents journaliers, les incidents de garde doivent être aisément liés aux diverses instances d'entités.	Visible
LB F5	F5.1 Les incidents ainsi que les demandes de travail sont liées aux instances d'entités. F5.2 La visualisation d'une instance d'entité permettra de trouver tous les incidents ainsi que toutes les demandes de travail liées.	Visible
LB F6	Les incidents encodés devront pouvoir être traités statistiquement	Visible
LB F7	Le S.I. doit permettre la recherche par mots clés.	Visible

Table 3-3

3.5.4. CONTRAINTES FONCTIONNELLES / NON FONCTIONNELLES AGORAGET

Identifiant	Contraintes fonctionnelles	
AGET F1	F1.1 Le S.I. doit récolter de manière automatique (sans intervention humaine) des informations mises à disposition par des applications ou des systèmes. F2.2 La récolte d'information se fera via l'exécution de programmes tiers (shell scripts, exécutables, fichiers .BAT...) qui retourneront de l'information textuelle. F 2.3 Les informations échangées entre AGORAGET et les programmes tiers seront structurées suivant un format qui reste à définir.	
AGET F2	La fréquence des récoltes d'informations doit être planifiable par l'outil de planification de Mobistar.	
AGET F3	La récolte des informations doit pouvoir se faire de manière sécurisée pour les systèmes critiques.	
AGET F4	Le S.I. doit permettre d'exécuter des traitements sur les informations récoltées.	
AGET F5	Le S.I. doit permettre d'organiser l'information ainsi récoltée pour permettre son exploitation dans l'application AGORA.	
Identifiant	Contraintes non fonctionnelles	
AGET NF1	Le S.I. doit pouvoir récolter les informations sur tout système UNIX (SUN, COMPAQ, HP) et sur les systèmes Microsoft NT.	
AGET NF2	Le S.I. doit utiliser pour la planification l'outil standard de Mobistar: Unicenter workload de Computer Associates.	

Table 3-4

4. Evaluation des solutions

Ce chapitre présente en premier lieu la recherche d'outils que nous avons effectuée ainsi qu'une description critique des types d'outils disponibles sur le marché. Nous poursuivons ensuite en précisant les choix que nous avons effectués et les orientations qui en découlent.

4.1. INTRODUCTION

Les contraintes fonctionnelles ont été divisées précédemment en trois parties pour des raisons de priorités. D'un point de vue fonctionnel, ces trois catégories peuvent se regrouper en:

- AGORAGET
- AGORAWATCH & AGORALOG

C'est sur base de ces 2 groupes que des solutions on été recherchées.

Nous avons en premier lieu recherché des outils intégrés permettant de répondre à nos besoins. En effet, la politique IT de Mobistar donne presque toujours sa préférence à l'achat d'outil existant sur le marché par rapport à une solution dévelopée spécifiquement. Mobistar recherche également des fournisseurs capables de délivrer un support de haute qualité.

4.2. OUTILS

La recherche d'outils répondant à nos contraintes a été en partie réalisée grâce à une licence d'accès à la base d'information du Gartner Group que Mobistar possède.

Les outils de gestion de contenu peuvent se regrouper en différentes catégories que nous allons brièvement décrire.

4.2.1. IDM (INTEGRATED DOCUMENT MANAGEMENT)

Les produits IDM gèrent des documents ainsi que d'autres formes de contenu pour un usage interne à la société. Ils permettent aux utilisateurs d'utiliser leurs outils familiers (traitement de texte...) et sauvegardent le contenu créé par ces applications dans un "IDM

Repository". Par l'adjonction de modules spécifiques, ces produits permettent de scanner des informations papier et de les stocker dans le "repository".

Ces outils étaient généralement des outils d'imaging. Permettant de scanner de grandes quantités de documents et de les intégrer dans un "workflow". Ces outils commencent à intégrer des notions de gestion de contenu structuré, d'autres encore commencent à incorporer des possibilités de "web publishing" mais cela reste assez marginal.

Les outils IDM ont pour but principal:

- La gestion de contenu à usage interne.
- Le niveau le plus bas de granularité est généralement le document
- Le contenu est avant tout statique

Il existe sur le marché toute une série d'outils IDM dont les plus connus sont:

- FileNet (http://www.filenet.com)
- Documentum (http://www.documentum.com)
- IntraNet Solutions (http://www.intranetsolution.com)

4.2.2. WCM (WEB CONTENT MANAGEMENT)

Les produits WCM sont essentiellement destinés à la mise à disposition de grandes quantités d'informations dynamiques et à destination d'un grand nombre d'utilisateurs. Ils utilisent des techniques comme le caching de contenu et la réplication. La plupart supporte un accès venant d'assistants digitaux personnels ainsi que d'équipements "wireless". La personnalisation de l'information en fonction de l'utilisateur est également une des fonctionnalités principales de ce genre d'outils. Des templates sont utilisés ici pour standardiser l'affichage mais pas pour

structurer l'information. Ces outils s'approchent plus de la gestion d'un site web à grande échelle que de la gestion d'informations structurées.

Le gartner group estime la mise en place d'une telle solution (licences et consultance de mise en place) à 500.000$.

Les outils WCM ont pour but principal:

- La gestion de contenu dynamique.
- La visualisation du contenu à partir de différents équipements.
- La personnalisation de l'information.
- Le haut débit (en millions d'utilisateurs).
- De structurer le layout de l'information.

Il existe sur le marché toute une série d'outils WCM dont les plus connus sont:

- Vignette (http://www.vignette.com)
- Interwoven (http://www.interwoven.com)
- BroadVision (http://www.braodvision.com)

4.2.3. EIP (ENTERPRISE INFORMATION PORTAL)

Les produits EIP sont essentiellement destinés à intégrer des données informatiques de types multiples, de formats multiples et provenant de multiples sources. De ce fait, l'utilisateur peut accéder à une information cohérente et utile concernant un sujet bien défini.

La gestion de l'information elle-même se fait généralement en dehors de l'EIP lui-même. Les sources d'information peuvent être multiples. Elles peuvent provenir d'Internet (information publique ou souscrite), de fichiers, d'applications, de partenaires...

Les outils EIP ont pour but principal:

- L'intégration d'information provenant de sources multiples

- Le filtrage de l'information.
- Le formatage de l'information
- L'indexation
- Le haut débit (en millions d'utilisateurs).
- La granularité de l'information au niveau du composant.

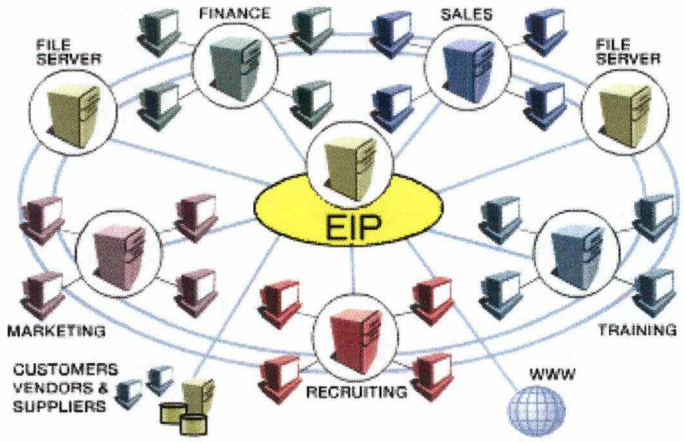


Schéma 1

Il existe sur le marché toute une série d'outils EIP dont les plus connus sont:

- datachannel (http://www.datachanel.com)
- IBM (http://www-4.ibm.com/software/data/eip)
- Oracle (http://www.oracle.com/portals/)

Voici un exemple de portail:

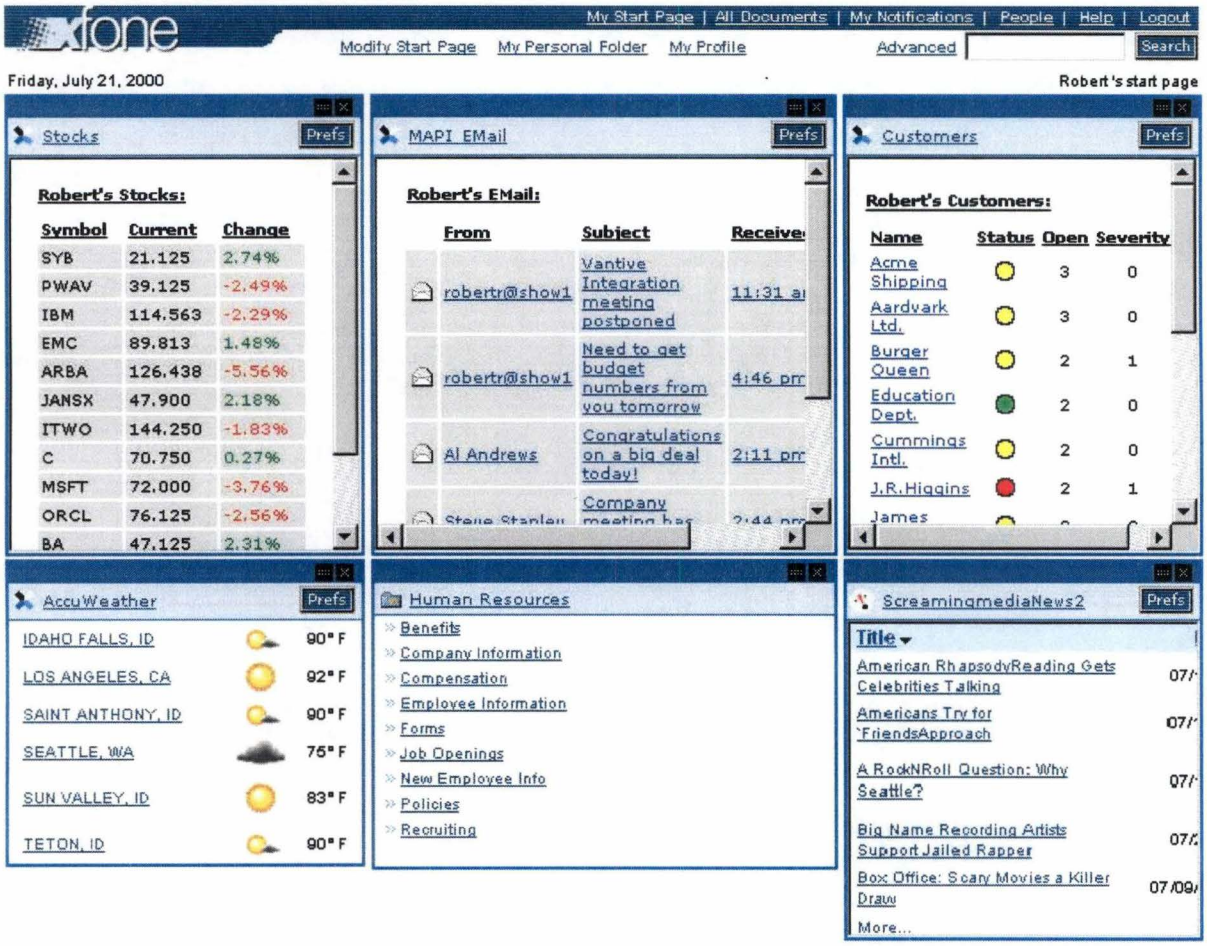


Schéma 2

4.2.4. WEB AUTHORING TOOLS

D'après le W3C, la définition de "Web authoring tools" serait: l'ensemble des outils dont le but est de créer des documents destinés à être publiés sur le web. Les "Web authoring tools" incluent généralement:

- Les outils d'édition destiné à produire du contenu pour le web (outils WYSIWYG HTML et éditeurs XML).

- Les outils permettant de sauvegarder de l'information dans un format Internet (traitement de texte permettant de sauvegarder en format HTML par exemple).
- Les outils permettant de créer du contenu multimédia à destination d'Internet.
- Les outils destinés à la gestion ou à la publication de sites Internet. Cela inclus les outils qui génèrent du contenu automatiquement à partir de base de données.
- Les outils permettant la gestion de la visualisation des informations (outils de gestion de style sheet).

Il existe sur le marché toute une série d'outils de type "authoring tool" dont les plus connus sont:

- Dreamweaver
- FrontPage
- Homesite
- Hotdog
- PageMill
- Netscape composer
- Microsoft word
- WebEdit

Ces outils ont pour vocation la création, l'édition et la mise à disposition d'informations dans un format compatible avec l'Internet. Ce sont essentiellement de petites applications permettant de créer des pages web en structurant un minimum l'affichage. La structure de l'information n'est pas du tout du ressort de ces outils.

4.3. CHOIX FINAL

En partant à la recherche d'un outil correspondant à nos besoins, nous pensions sérieusement trouver une série d'outils qui pourraient correspondre à nos besoins. Nous les aurions alors comparés en fonction de nos contraintes pour choisir le meilleur.

Il a fallu pourtant que nous nous rendions à l'évidence, **aucun outil intégré ne peut répondre à nos contraintes fonctionnelles principales (AWAT F1.1 ; AWAT F2 ; AWAT NF4 - cfr 3.5.2)**. De plus, pour chaque catégorie d'outils, toute une série de raisons supplémentaires nous permettent de rejeter leur choix.

Les outils IDM

Nous avons écarté les outils IDM pour les raisons supplémentaires suivantes:

- La granularité minimale est le document alors que nous désirons structurer le contenu des documents.
- La manipulation des informations se fait généralement à travers divers outils à installer sur le poste de l'utilisateur et non à travers une interface web.
- Mobistar possède déjà un outil de type IDM dont le but est de gérer de très gros volumes de documents dans certains de nos départements orientés "clients". Cet outil possède les défauts énumérés plus haut. Il ne peut donc être utilisé et il nous serait impossible de faire entrer un autre outil "IDM" chez Mobistar.
- Les outils IDM ont beaucoup de fonctionnalités qui ne nous intéressent pas (scanning de grandes quantité de documents, workflow...) mais qui portent à conséquence dans le prix final.

Les outils WCM

Nous avons écarté les outils WCM pour les raisons supplémentaires suivantes:

- Les plus complets d'entre eux sont essentiellement destinés à la mise à dispotion de l'information à destination de l'Internet (gestion du "look" de site web) ce qui n'est pas du tout notre but.
- Les outils WCM ont beaucoup de fonctionnalités qui ne nous intéressent pas (haut volume de transactions possible) mais qui portent à conséquence dans le prix final.

Les outils EIP

Nous avons écarté les outils EIP pour la raison supplémentaire suivante:

- Ces outils ne permettent que d'agréger de l'information créée par d'autres outils ce qui n'est pas du tout le but recherché ici.

Les outils de type "Web authoring tools"

Nous avons écarté les outils de type "Web authoring tools" pour les raisons supplémentaires suivantes:

- Les plus complets d'entre eux sont essentiellement destinés à la création de fichiers HTML qui seront ensuite intégrés dans un serveur Web.
- L'équipe système gère actuellement sa documentation avec un des outils de ce type: FrontPage. Elle n'est pas satisfaite car comme tous les outils de ce type, il se concentre sur le "layout" et jamais sur la structure des données.

4.4. CONCLUSION

La solution doit donc être une solution développée spécifiquement pour Mobistar. Les chapitres suivants seront constitués de l'analyse ainsi que de l'architecture de l'application. Cependant, les orientations générales peuvent déjà être définie car ce sont des standards chez Mobistar pour tout nouveau développement.

4.5. ORIENTATIONS

4.5.1. FRAMEWORK

Ce développement doit suivre les standards Mobistar en la matière pour toute nouvelle application:

- Langage de développement: Java
- Framework: J2EE
- Environnement de développement: BEA WebGain
- Environnement de déploiement: BEA WebLogic E-Business Platform
- Système opératoire de production: Sun Solaris
- Sytème opératoire de développement: Windows NT / Sun Solaris

4.5.2. DESIGN

Nous avons observé lors de nos investigations à propos des outils de gestion de l'information qu'ils convergent tous vers une structuration de l'information plus précise que le simple document. Cette convergence n'en est encore qu'à ses débuts et est, dans presque tous les cas, basée sur un standard émergeant: XML.

XML est en passe de devenir un standard incontournable. XML permet en effet à la fois le formatage de documents à des fins d'échange entre applications, la communication de messages entre des systèmes et

4.4. CONCLUSION

La solution doit donc être une solution développée spécifiquement pour Mobistar. Les chapitres suivants seront constitués de l'analyse ainsi que de l'architecture de l'application. Cependant, les orientations générales peuvent déjà être définie car ce sont des standards chez Mobistar pour tout nouveau développement.

4.5. ORIENTATIONS

4.5.1. FRAMEWORK

Ce développement doit suivre les standards Mobistar en la matière pour toute nouvelle application:

- Langage de développement: Java
- Framework: J2EE
- Environnement de développement: BEA WebGain
- Environnement de déploiement: BEA WebLogic E-Business Platform
- Système opératoire de production: Sun Solaris
- Sytème opératoire de développement: Windows NT / Sun Solaris

4.5.2. DESIGN

Nous avons observé lors de nos investigations à propos des outils de gestion de l'information qu'ils convergent tous vers une structuration de l'information plus précise que le simple document. Cette convergence n'en est encore qu'à ses débuts et est, dans presque tous les cas, basée sur un standard émergeant: XML.

XML est en passe de devenir un standard incontournable. XML permet en effet à la fois le formatage de documents à des fins d'échange entre applications, la communication de messages entre des systèmes et

même l'enregistrement de données dans une banque de données. De plus, un tel document message ou fichier contient encore en lui même des informations sur la structuration des uns et des autres, de sorte qu'un autre système puisse, la cas échéant, interpréter l'information tout à fait automatiquement.

Enfin, Mobistar utilise XML dans un nombre croissant d'application.

XML sera donc à la base de l'architecture de l'outil que nous allons faire développer.

5. Cas d'utilisation

Ce chapitre décrit les acteurs concernés par le S.I. ainsi que les cas d'utilisation qui correspondent aux auditions que nous avons menées auprès des différentes équipes du département IOP. En fin de chapitre se trouve un tableau synthétisant la correspondance entre les contraintes fonctionnelles et les cas d'utilisation.

5.1. LES ACTEURS

5.1.1. L'UTILISATEUR

Un utilisateur accède au S.I. pour visualiser les informations et en modifier le contenu s'il en a les droits. Les informations visualisantes sont les instances des templates ainsi que les rapports.

5.1.2. L'ADMINISTRATEUR

L'administrateur est la personne qui administre le S.I. En plus des privilèges utilisateurs, il peut créer de nouveaux templates, manipuler la structure des templates existants ainsi que créer de nouveaux rapports.

5.1.3. LE SCHEDULER

Chez Mobistar le scheduler est *unicenter work load*. Le scheduler demande au S.I. l'exécution de programmes sur des S.I. externes. Le S.I. lance l'exécution de ces programmes sur les S.I. externes. Ces programmes sont destinés à récolter de l'information sur le S.I. externe sur lequel il a été lancé. Le S.I. récupère l'information et la stocke de manière structurée.

5.1.4. LES SI EXTERNES

Chez Mobistar, les S.I. externes sont des serveurs de type UNIX ou NT.

5.2. USE CASE : PRINCIPAL

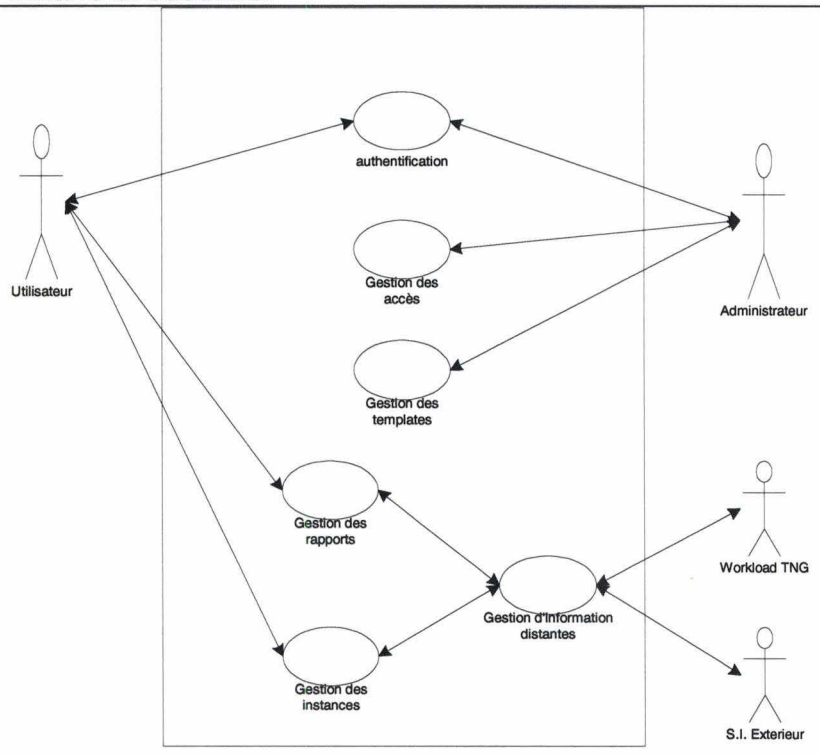
Acteurs	Utilisateur, administrateur et le scheduler
Description:	Ce use case commence quand un utilisateur, un administrateur ou le scheduler désire utiliser le S.I. pour structurer, manipuler, récolter, visualiser l'information.
Type	Primaire et essentiel
Diagramme	

Schéma 3

Flux normal	
Action d'un utilisateur, d'un administrateur ou le scheduler	Action du SI
1. L'acteur accède au S.I.	
	2. le S.I. authentifie l'administrateur, l'utilisateur ou le scheduler: ***Use case :*** Authentification ***ch. 5.3*** et affiche les opérations réalisables par l'acteur.
3. L'acteur peut choisir une opération: 3.1 Si l'acteur est un administrateur, il peut choisir: • Créer un utilisateur: ***Use case : Création d'un*** utilisateur***utilisateur ch. 5.4*** • Rechercher, visualiser, modifier, supprimer un utilisateur: ***Use case : Recherche / visualisation / modification d'un utilisateur ch. 5.5*** • Créer une équipe: ***Use case : Création d'une équipe ch. 5.6*** • Rechercher, visualiser, modifier une équipe: ***Use case : Recherche / visualisation / modification d'une*** équipe ***équipe ch. 5.7*** • Création d'un template: ***Use case : Création d'un template ch. 5.8*** • Recherche, visualisation d'un template: ***Use case : Recherche / visualisation / modification d'un template ch. 5.9*** • Créer la structure d'un rapport: ***Use case : Création d'un rapport ch. 5.10*** • Rechercher, visualiser, modifier la structure d'un rapport: ***Use case : Rechercher, visualiser, modifier un rapport ch. 5.11***	

3.2. Si l'acteur est un utilisateur, il peut choisir: • Créer une instance:***Use case : Création d'une*** instance ***5.12*** • Rechercher, visualiser, modifier une instance: ***Use case : Recherche / visualisation / modification d'une instance ch. 5.13*** • Exécuter un rapport: ***Use case : Exécution d'un rapport ch. 5.14*** • Rechercher de l'information***: Use case : Recherche d'information ch. 5.16*** 3.3. Si l'acteur est le scheduler, il peut choisir: • Exécuter une collecte d'information: ***Use case : Exécution d'une collecte*** d'information ***ch. 5.15***	
Flux Alternatif	
Point 2. Si l'authentification est refusée, l'acteur se retrouve alors au point 1	

5.3. USE CASE : AUTHENTIFICATION	
Acteurs	Administrateur, utilisateur, le scheduler
But	Authentification d'un administrateur, d'un utilisateur ou du scheduler
Description	Ce use case commence quand l'acteur désire accéder au S.I.. Pour accéder aux données du S.I. , un acteur doit être authentifié.
Type	Primaire et essentiel

Flux normal	
Action de l'acteur	Action du SI
	1. Le SI demande un nom et un mot de passe
2. L'acteur encode les informations et les soumet au S.I. .	
	3. Le SI authentifie l'acteur en tant qu' administrateur, utilisateur ou scheduler.
Flux Alternatif	
Point 3. Si les informations ne sont pas correctes l'acteur n'est pas authentifié et il se retrouve alors au point 1	

5.4. USE CASE : CRÉATION D'UN UTILISATEUR

Acteurs	Administrateur
But	Créer un utilisateur
Description	Ce use case commence quand un administrateur authentifié désire gérer les accès au S.I.. Pour qu'un utilisateur puisse accéder au S.I., il faut qu'il ait été créé par l'administrateur.
Type	Primaire et essentiel

Flux normal	
Action de l'administrateur	Action du SI
	1. Le S.I. demande les informations concernant l'utilisateur (nom, prénom, équipe, description, actif). Seul un utilisateur actif peut accéder au S.I. Un utilisateur non actif est un utilisateur qui ne peut plus accéder au S.I. mais dont on doit garder trace.
2. L'administrateur encode les informations demandées.	
	3. Le S.I. crée l'utilisateur.
Flux Alternatif	
1. Si le nom existe déjà, afficher une erreur 2. Si l'équipe n'existe pas, afficher une erreur	

5.5. USE CASE : RECHERCHE / VISUALISATION / MODIFICATION D'UN UTILISATEUR

Acteurs	Administrateur
But	Manipulation des données relatives à un utilisateur du S.I.
Description	Ce use case commence quand l'administrateur désire rechercher / visualiser / modifier les données relatives aux utilisateurs ou empêcher un utilisateur de se connecter au S.I. (le rendre inactif).
Type	Primaire et essentiel

Flux normal	
Action de l'administrateur	Action du SI
	1. Le S.I. propose une recherche d'utilisateur suivant divers critères: nom, prénom, équipe, actif(oui/non)
2. L'administrateur demande la recherche en fonction des critères	
	3. Le S.I. affiche une liste des utilisateurs répondant aux critères
4. L'administrateur sélectionne un utilisateur	
	5. Le S.I. affiche les données relatives à l'utilisateur sélectionné
6. L'administrateur peut modifier les données de l'utilisateur. Si la valeur "actif (oui/non)"est mise à non, l'utilisateur ne pourra plus se connecter.	
	7. Le S.I. met à jour les informations de l'utilisateur.
Flux Alternatif	

5.6. USE CASE : CRÉATION D'UNE ÉQUIPE

Acteurs	Administrateur
But	Créer une équipe
Description	Ce use case commence quand un administrateur authentifié désire gérer les accès au S.I. Les droits d'accès aux informations du S.I. sont basés sur les équipes. Pour pouvoir manipuler une instance, un utilisateur doit appartenir à une équipe et son équipe doit avoir accès à cette instance.
Type	Primaire et essentiel

Flux normal	
Action de l'administrateur	Action du SI
	1. Le S.I. demande les informations concernant l'équipe (nom, description).
2. L'administrateur encode les informations demandées.	
	3. Le S.I. crée l'équipe.
Flux Alternatif	
1. Si le nom existe déjà, afficher une erreur	

5.7. USE CASE : RECHERCHE / VISUALISATION / MODIFICATION D'UNE ÉQUIPE

Acteurs	Administrateur
But	Manipulation des données relatives à une équipe.
Description	Ce use case commence quand l'administrateur désire rechercher / visualiser / modifier les données relatives aux équipes ou empêcher une équipe de se connecter au S.I. (le supprimer logiquement).
Type	Primaire et essentiel

Flux normal	
Action de l'administrateur	Action du SI
	1. Le S.I. affiche une liste des équipes.
2. L'administrateur sélectionne une équipe	
	3. Le S.I. affiche les données relatives à l'équipe sélectionnée
4. L'administrateur peut modifier les données de l'utilisateur (nom, description).	
	5. Le S.I. met à jour les informations de l'équipe.
Flux Alternatif	

5.8. USE CASE : CRÉATION D'UN TEMPLATE

Acteurs	Administrateur
But	Création d'un template
Description	Ce use case commence quand un administrateur désire créer un template. Un template définit la structure de l'information. Les utilisateurs utiliseront ces templates pour créer des instances.
Type	Primaire et essentiel

Flux normal	
Action de l'administrateur	Action du SI
	1. Le S.I. demande les informations concernant le template (nom, description).
2. L'administrateur encode les informations demandées.	
	3. Le S.I. crée le template avec les éléments minimum constituant un template: - Un élément de type "Nom" - Un élément de type "Sécurité"
Flux Alternatif	
1. Si le nom existe déjà, afficher une erreur	

5.9. USE CASE : RECHERCHE / VISUALISATION / MODIFICATION D'UN TEMPLATE

Acteurs	Administrateur
But	Rechercher, visualiser et modifier un template.
Description	Ce use case commence quand l'administrateur désire modifier la structure d'un template.
Type	Primaire et essentiel

Flux normal	
Action de l'administrateur	Action du SI
	1. Le S.I. affiche une liste des templates.
2. L'administrateur sélectionne un template	
	3. Le S.I. affiche les éléments constituant le template sélectionné. 4. Le S.I. propose l'insertion d'un élément dans le template. Un élément peut être: - Une zone de lien (de type is depending; depends; generate; is generated by ; point to ; is pointing to) - Une liste d'éléments prédéfinis.
5. L'administrateur peut choisir d'insérer ces éléments dans le template. Il peut leur donner un ordre. Il peut leur ajouter les attributs de sécurité à chaque élément.	
	6 Le S.I. sauvegarde les modifications
Flux alternatif	

5.10. USE CASE : CRÉATION D'UN RAPPORT

Acteurs	Administrateur
But	Création d'un rapport
Description	Ce use case commence quand l'administrateur désire créer des rapports. Ceux-ci se basent sur les données des templates.
Type	Primaire et essentiel

Flux normal	
Action de l'administrateur	Action du SI
	1. Le S.I. demande les informations concernant le rapport (nom, description).
2. L'administrateur encode les informations demandées.	
	3. Le S.I. crée le rapport.
Flux Alternatif	

5.11. USE CASE : RECHERCHER, VISUALISER, MODIFIER UN RAPPORT

Acteurs	Administrateur
But	Rechercher, visualiser et modifier un rapport.
Description	Ce use case commence quand l'administrateur désire modifier la structure d'un rapport.
Type	Primaire et essentiel

Flux normal	
Action de l'administrateur	Action du SI
	1. Le S.I. affiche une liste des rapports.
2. L'administrateur sélectionne un rapport	
	3. Le S.I. affiche les éléments constituant le rapport sélectionné. 4. Le S.I. propose l'insertion d'un élément dans le rapport. Un élément peut être tout élément constitutif d'un template.
5. L'administrateur peut choisir d'insérer ces éléments dans le template. Il choisit des éléments parmi les templates définis. Il peut lier ces éléments entre eux par des opérateurs.	
	6 Le S.I. sauvegarde les modifications.
Flux Alternatif	

5.12. USE CASE : CRÉATION D'UNE INSTANCE

Acteurs	Utilisateur.
But	Création d'une instance.
Description	Ce use case commence quand un utilisateur désire créer une instance d'un template.
Type	Primaire et essentiel .

Flux normal	
Action de l'utilisateur	Action du SI
	1. Le S.I. affiche une liste des templates.
2. L'utilisateur sélectionne un template.	
	3. Le S.I. affiche la liste des instances que l'utilisateur a le droit de visualiser
4. L'utilisateur demande la création d'une nouvelle instance.	
	5. Le S.I. demande les informations concernant l'instance (le nom).
6. L'utilisateur encode les informations demandées.	
	7. Le S.I. crée l'instance avec la structure minimum imposée par le template.
Flux Alternatif	
9. Si le nom existe déjà, afficher une erreur	

5.13. USE CASE : RECHERCHE / VISUALISATION / MODIFICATION D'UNE INSTANCE

Acteurs	Utilisateur.
But	Rechercher, visualiser et modifier les données d'une instance.
Description	Ce use case commence quand l'utilisateur désire rechercher, visualiser et modifier les informations d'une instance.
Type	Primaire et essentiel.

Flux normal	
Action de l'administrateur	Action du SI
	1. Le S.I. affiche une liste des templates que l'utilisateur peut visualiser.
2. L'utilisateur sélectionne un template.	
	3. Le S.I. affiche la liste des instances que l'utilisateur a le droit de visualiser
4. L'utilisateur choisit une instance.	
	5. Le S.I. affiche les informations de l'instance
6. L'utilisateur peut modifier les données de l'instance en ajoutant, modifiant, supprimant des informations dans l'instance. Ces manipulations doivent respecter les manipulations définies par le template associé.	
	7. Le S.I. stocke les modifications
Flux alternatif	

5.14. USE CASE : EXÉCUTION D'UN RAPPORT	
Acteurs	Utilisateur.
But	Exécuter un rapport.
Description	Ce use case commence quand un utilisateur désire exécuter un rapport.
Type	Primaire et essentiel.

Flux normal	
Action de l'utilisateur	Action du SI
	1. Le S.I. affiche une liste des rapports.
2. L'utilisateur sélectionne un rapport	
	3. Le S.I. exécute le rapport et affiche les résultats. 4. Le S.I. propose l'impression du rapport.
5. L'utilisateur peut, s'il le désire, imprimer le rapport.	
	6. Le S.I. sauvegarde les modifications
Flux Alternatif	

5.15. USE CASE : EXÉCUTION D'UNE COLLECTE D'INFORMATIONS

Acteurs	Scheduler, des S.I. externes.
But	Exécuter une collecte d'informations.
Description	Un scheduler peut demander l'exécution d'une collecte d'informations.
Type	Primaire et essentiel.

Flux normal	
Action du scheduler, des S.I. externes	Action du SI
1. Le scheduler demande l'exécution d'une collecte d'informations.	
	2. Le S.I. demande l'exécution de la collecte d'informations aux S.I. externes. L'échange d'information entre le S.I. et les S.I. externes critiques doit être sécurisé.
3. Les S.I. externes exécutent la collecte d'informations et renvoient les informations au S.I.	
	4. Le S.I. incorpore le résultat de la collecte dans les instances d'un template.
Flux Alternatif	

5.16. USE CASE : RECHERCHE D'INFORMATIONS

Acteurs	L'utilisateur.
But	Exécuter une recherche d'informations.
Description	Ce use case commence quand un utilisateur désire faire une recherche d'informations.
Type	Primaire et essentiel.

Flux normal	
Action de l'utilisateur	Action du SI
1. L'utilisateur demande une recherche en fournissant un ou plusieurs mots clés.	
	2. Le S.I. recherche ce ou ces mots clés à travers toutes les instances contenues dans sa base de données. Il retourne à l'utilisateur une liste d'instances à laquelle l'utilisateur a accès et contenant le ou les mots clés.
3.L'utilisateur se retrouve au point 4 du use case:	
Use case : Recherche / visualisation / modification d'une instance	
ch. 0	
Flux Alternatif	

5.16.1. TABLEAU CONTRAINTES / USE CASE

Contraintes	USE CASE														
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
AWATF1	X										X	X	X		
AWATF2	X						X	X	X	X					
AWATF3	X						X	X	X	X					
AWATF4	X	X	X	X	X	X									
AWATF5	X	X	X	X	X	X									
AWATF6	X		X	X	X	X									
AWATF7	X						X	X							
AWATF8	X						X	X							
AWATF10	X						X	X							
AWATF13	X								X	X			X		
LB F1	X										X	X			
LB F2	X										X	X			
LB F3	X										X	X			
LB F4	X										X	X			
LB F5	X										X	X	X		
LB F6	X												X		
LB F7	X														X
AGET F1	X													X	
AGET F2	X													X	
AGET F3	X													X	
AGET F4	X													X	
AGET F5	X													X	
AWATF9 AWATF11 AWATF12 AWATF14 AWATF15	Use case non visible														

6. Fonctions

Ce chapitre décrit les fonctions déduites des cas d'utilisation du chapitre précédent.

6.1. FONCTIONS DE GESTION DES ACCÈS

6.1.1. FONCTION DE LOGIN

Arguments

- Le S.I.
- Nom de l'utilisateur.
- Mot de passe de l'utilisateur.

Résultats

Le S.I.'

Pré-conditions

L'utilisateur dispose d'un nom d'utilisateur et d'un mot de passe connus du système.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I..
- L'utilisateur est authentifié par le système.
- S'il est administrateur, l'utilisateur est alors un administrateur authentifié par le système.

6.1.2. FONCTION DE CRÉATION D'UN UTILISATEUR

Arguments

- Le S.I.
- Le nom de l'utilisateur à créer.
- La liste des équipes auxquelles l'utilisateur appartient.
- Les informations concernant l'utilisateur.

Résultats

Le S.I.'

Pré-conditions

- L'utilisateur est un administrateur authentifié.
- Le nom de l'utilisateur à créer n'existe pas dans le S.I..

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.. De plus, il contient les informations du nouvel utilisateur.

6.1.3. FONCTION DE RECHERCHE D'UN UTILISATEUR

Arguments

- Le S.I.
- le nom de l'utilisateur.
- Les critères de recherche d'un utilisateur.

Résultats

- Le S.I.'
- Les utilisateurs répondant aux critères de recherche et que l'utilisateur a la permission de visualiser.

Pré-conditions

- L'utilisateur est un administrateur authentifié.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.

6.1.4. FONCTION DE MODIFICATION D'UN UTILISATEUR

Arguments

- Le S.I.
- Le nom de l'utilisateur.
- Les informations concernant l'utilisateur.

Résultats

- Le S.I.'

Pré-conditions

- L'utilisateur est un administrateur authentifié.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.. De plus, il contient les informations modifiées de l'utilisateur.

6.1.5. FONCTION DE CRÉATION D'UNE ÉQUIPE

Arguments

- Le S.I.
- Le nom de l'équipe à créer.
- Les informations concernant l'équipe.

Résultats

- Le S.I.'

Pré-conditions

- L'utilisateur est un administrateur authentifié.
- Le nom de l'équipe à créer n'existe pas dans le S.I.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.. De plus, il contient les informations de la nouvelle équipe.

6.1.6. FONCTION DE RECHERCHE D'UNE ÉQUIPE

Arguments

- Le S.I.
- Les critères de recherche d'une équipe.
- Le nom de l'utilisateur.

Résultats

- Le S.I.'
- Les équipes répondant aux critères de recherche et que l'utilisateur a la permission de visualiser.

Pré-conditions

- L'utilisateur est authentifié.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I..

6.1.7. FONCTION DE MODIFICATION D'UNE ÉQUIPE

Arguments

- Le S.I.
- Le nom de l'équipe.
- Les informations concernant l'équipe.

Résultats

- Le S.I.'

Pré-conditions

- L'utilisateur est un administrateur authentifié.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.. De plus, il contient les informations modifiées de l'équipe.

6.2. FONCTIONS DE GESTION DES TEMPLATES

6.2.1. FONCTION DE CRÉATION D'UN TEMPLATE

Arguments

- Le S.I.
- Le nom du template.
- Les informations concernant le template.

Résultats

- Le S.I.'

Pré-conditions

- L'utilisateur est un administrateur authentifié.
- Le nom du template à créer n'existe pas dans le S.I.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.. De plus, il contient les informations du nouveau template.

6.2.2. FONCTION DE RECHERCHE DE TEMPLATES

Arguments

- Le S.I.
- Les critères de recherche d'un template.
- Le nom de l'utilisateur.

Résultats

- Le S.I.'
- Les templates répondant aux critères de recherche que l'utilisateur est autorisé à visualiser.

Pré-conditions

- L'utilisateur est authentifié.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.

6.2.3. FONCTION D'AJOUT / SUPRESSION D'UN ÉLÉMENT DANS UN TEMPLATE

Arguments

- Le S.I.
- Le nom du template.
- L'opération à effectuer (ajouter/supprimer).
- L'endroit où l'opération doit s'effectuer dans le template.
- L'élément à insérer dans le template (si l'opération est ajouter).

Résultats

- Le S.I.'

Pré-conditions

- L'utilisateur est un administrateur authentifié.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.. De plus, il contient les informations modifiées du template.

6.2.4. FONCTION D'AJOUT / SUPRESSION D'ATTRIBUTS SÉCURITÉ À UN ÉLÉMENT D'UN TEMPLATE

Arguments

- Le S.I.

- Le nom du template.
- L'opération à effectuer (ajouter/supprimer).
- l'endroit où l'opération doit s'effectuer dans le template.

Résultats

- Le S.I.'

Pré-conditions

- L'utilisateur est un administrateur authentifié.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.. De plus, il contient les informations modifiées du template.

6.3. FONCTIONS DE GESTION DES INSTANCES

6.3.1. FONCTION DE CRÉATION D'UNE INSTANCE

Arguments

- Le S.I.
- Le nom du template.
- Le nom de l'instance.
- Le nom de l'utilisateur.

Résultats

- Le S.I.'

Pré-conditions

- L'utilisateur est authentifié et autorisé à créer des instances pour ce template.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.. De plus, il contient les informations de la nouvelle instance.

6.3.2. FONCTION DE RECHERCHE D'INSTANCES D'UN TEMPLATE

Arguments

- Le S.I.
- Le nom du template.
- Les critères de recherche d'un template.
- Le nom de l'utilisateur.

Résultats

- Le S.I.'
- Les instances répondant aux critères de recherche que l'utilisateur a l'autorisation de visualiser.

Pré-conditions

- L'utilisateur est authentifié.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.

6.3.3. FONCTION DE RECHERCHE DES INFORMATIONS D'UNE INSTANCE

Arguments

- Le S.I.
- Le nom du template.
- Le nom de l'instance.
- Le nom de l'utilisateur.

Résultats

- Le S.I.'
- Les informations de l'instance du template.

Pré-conditions

- L'utilisateur est authentifié.
- L'utilisateur est autorisé à visualiser les informations de cette instance de ce template.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.

6.3.4. FONCTION DE MODIFICATION DES INFORMATIONS D'UN ÉLÉMENT D'UNE INSTANCE

Arguments

- Le S.I.
- Le nom du template.
- Le nom de l'instance.
- Le nom de l'élément.
- Les nouvelles informations de l'élément.
- Le nom de l'utilisateur.

Résultats

- Le S.I.'

Pré-conditions

- L'utilisateur est authentifié.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.. De plus, il contient les informations modifiées de l'élément.

6.4. FONCTIONS DE GESTION DES RAPPORTS

6.4.1. FONCTION DE RECHERCHE DE RAPPORTS

Arguments

- Le S.I.
- Les critères de recherche d'un rapport.
- Le nom de l'utilisateur.

Résultats

- Le S.I.'
- Les templates répondant aux critères de recherche.

Pré-conditions

- L'utilisateur est authentifié.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.

6.4.2. FONCTION D'EXÉCUTION D'UN RAPPORT

Arguments

- Le S.I.
- Le nom du rapport.
- Le nom de l'utilisateur.

Résultats

- Le S.I.'
- Les informations du rapport.

Pré-conditions

- L'utilisateur est authentifié.
- L'utilisateur est autorisé à visualiser les informations de ce rapport.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I..

6.5. FONCTION DE GESTION DES COLLECTES D'INFORMATION

6.5.1. FONCTION D'EXÉCUTION D'UNE COLLECTE D'INFORMATIONS

Arguments

- Le S.I.
- Le nom de la collecte.

Résultats

- Le S.I.'

Pré-conditions

- L'utilisateur est un scheduler.

Post-conditions

- Le S.I.' contient les mêmes informations que le S.I.. De plus, il contient les informations de la collecte.

7. Schéma entité association

7.1. ENTITÉ ELEMENT

- Définition : contient les informations relatives à un élément simple ou complexe. Ces éléments peuvent être utilisés pour construire la structure d'un template.
- Attributs :
  - Element Name : Nom de l'élément
  - Default Min Occurrence : Valeur par défaut du nombre minimum d'occurrence d'un élément dans une structure d'un template.
  - Default Max Occurrence : Valeur par défaut du nombre maximum d'occurrence d'un élément dans une structure d'un template

7.1.1. SPÉCIALISATION DE L'ENTITÉ ELEMENT : SIMPLE

- Définition : contient les informations relatives aux éléments simple
- Attributs :
  - Type : type d'un élément
  - Default Value : liste de valeur par défaut d'un élément

7.1.2. SPÉCIALISATION DE L'ENTITÉ ELEMENT : COMPLEX

- Définition : contient les informations relatives aux éléments complexes. Un élément complexe est composé d'un ou plusieurs éléments qui sont eux-mêmes simples ou complexes.
- Attributs : /

7.2. ENTITÉ TEMPLATE

- Définition : contient les informations relatives aux templates
- Attributs :
  - Template Name : nom du template.
  - Creation Date : date de création du template.
  - Modification Date : data de dernière modification de la structure du template.

7.3. ENTITÉ ELEMENT TEMPLATE DEFINITION

- Définition : contient les informations relatives à la structure d'un template
- Attributs :
  - Element Template Id : est l'identifiant d'un élément de la structure du template.
  - Element : définition de l'élément associé au nœud de la structure du template
    - Type : Type de l'élément
    - Default Value : liste des valeurs possibles de l'élément.
    - Default Min Occurrence : nombre minimum d'occurrence d'un élément dans une structure d'un template.
    - Default Max Occurrence : nombre maximum d'occurrence d'un élément dans une structure d'un template.
  - Node : nœud dans la structure auquel est rattaché l'élément
    - Node Id : numéro du nœud
    - Prnt Node Id : parent du nœud de l'élément courant. Cela permet de définir l'arborescence de la structure qui compose le template.

7.4. ENTITÉ INSTANCE

- Définition : Une instance d'un template est un document conforme à la structure du template dont les éléments on étés remplacés par des données.
- Attributs :
  - Instance Id : Identifiant d'une instance d'un template.
  - Création date: Date de création de l'instance.
  - Modification date: Date de modification de l'instance.

7.4.1. SPÉCIALISATION DE L'ENTITÉ INSTANCE : NUMBER

- Définition : Permet de stocker un nombre.
- Attributs : /

7.4.2. SPÉCIALISATION DE L'ENTITÉ INSTANCE : CHAR

- Définition : Permet de stocker une chaîne de caractères.
- Attributs : /

7.4.3. SPÉCIALISATION DE L'ENTITÉ INSTANCE : HYPERLINK

- Définition : Permet de stocker un hyper lien.
- Attributs : /

7.4.4. SPÉCIALISATION DE L'ENTITÉ INSTANCE : BLOB

- Définition : Permet de stocker un blob.
- Attributs : /

7.4.5. SPÉCIALISATION DE L'ENTITÉ INSTANCE : OTHER

- Définition : Représente tous les autres types de données qu'il faudra supporter.
- Attributs : /

7.5. ENTITÉ STYLESHEET

- Définition : contient les informations relatives à la stylesheet.
- Attributs :
  - Style Sheet Name : nom de la stylesheet.
  - Creation Date : date de création de la stylesheet.
  - Modification Date : data de dernière modification de la structure de la stylesheet.

7.6. ENTITÉ ELEMENT STYLE SHEET DEFINITION

- Définition : contient les informations relatives à l'affichage et/ou d'impression d'un élément de la structure d'un template.
- Attributs :
  - Element Style Sheet Id : identifie un des éléments d'affichage et/ou d'impression de la stylesheet qui seront appliqués à l'élément d'un template.

7.6.1. SPÉCIALISATION DE L'ENTITÉ ELEMENT STYLE SHEET DEFINITION : "FONT"

- Définition : contient les informations relatives au "font" (police de caractère) qui vont être appliquées à un élément d'un template.
- Attributs : /

7.6.2. SPÉCIALISATION DE L'ENTITÉ ELEMENT STYLE SHEET DEFINITION : SHADE

- Définition : contient les informations relatives au "shade" (remplissage) qui vont être appliquées à un élément d'un template.
- Attributs : /

7.6.3. SPÉCIALISATION DE L'ENTITÉ ELEMENT STYLE SHEET DEFINITION : ALIGN

- Définition : contient les informations relatives à la mise en forme qui vont être appliquées à un élément d'un template.
- Attributs : /

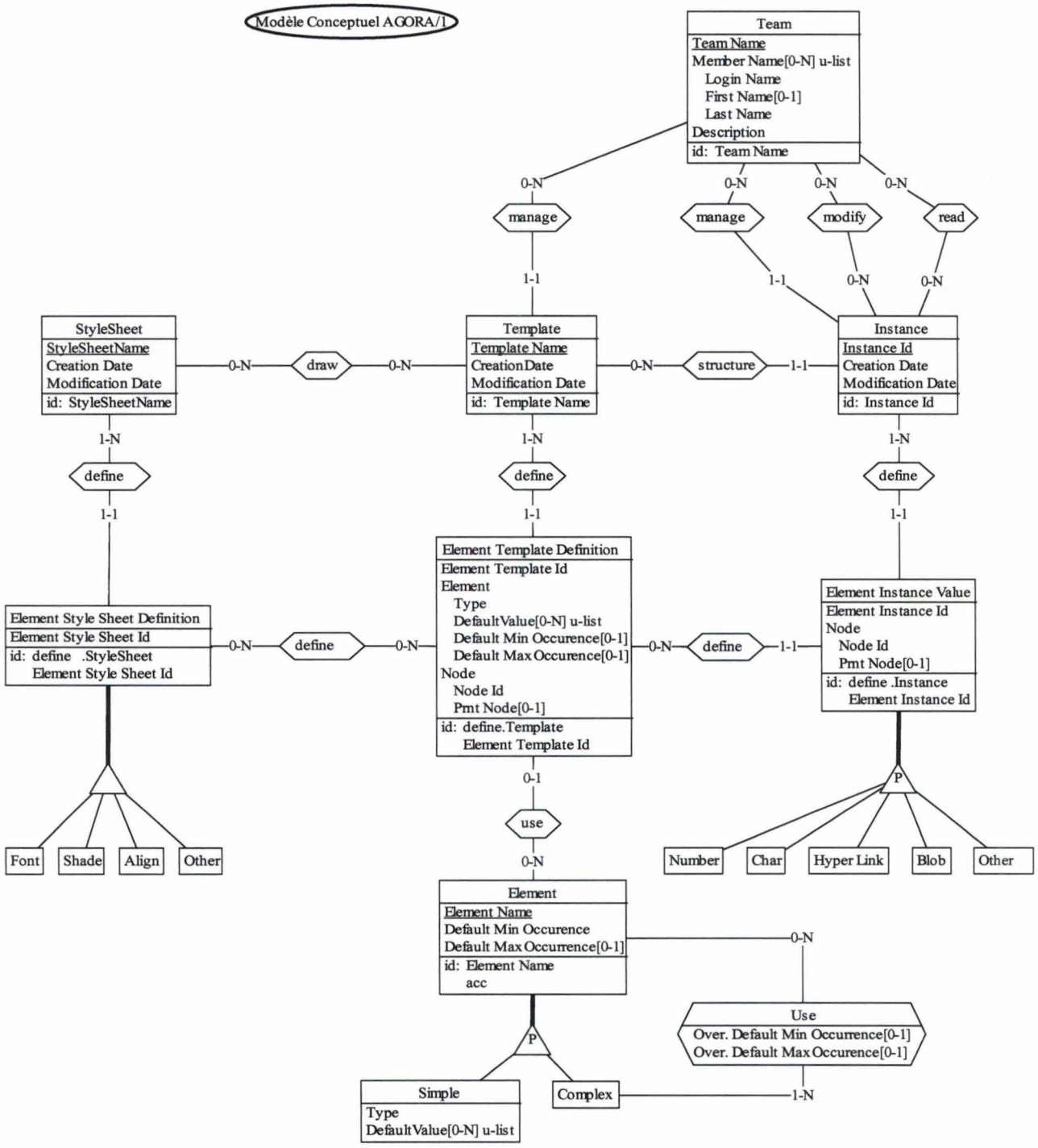
7.6.4. SPÉCIALISATION DE L'ENTITÉ ELEMENT STYLE SHEET DEFINITION : OTHER

- Définition : montre que toutes les possibilités n'ont pas encore été découvertes.
- Attributs :/

7.7. ENTITÉ TEAM

- Définition : contient la définition des équipes qui vont accéder le S.I.
- Attributs :
  - Team Name : nom de l'équipe
  - Descritpion: Texte décrivant le team
  - Member Name :liste de membre qui font partie de l'équipe.
  - Login Name : nom faisant le lien avec l'outil d'authentification.
  - First Name : prénom d'un membre
  - Last Name : nom d'un membre

7.8. SCHÉMA ENTITÉ ASSOCIATION



8. Conception Globale

8.1. INTRODUCTION

L'architecture globale doit couvrir les deux grandes catégories fonctionnelles du mémoire :

- L'acquisition automatique des données par un système de soumission de jobs, repris sous le nom **AGORAGET**.
- La gestion des informations du service IOP, repris sous le nom **AGORAWATCH**.

Dans le cadre de ces applications on a choisi d'utiliser un développement basé sur une architecture n-tier où chaque tier représente une couche application qui ne peut communiquer avec les autres couches qu'au travers d'interfaces prédéfinies qui fournissent des services. Chaque couche peut être totalement réécrite, tant qu'elle continue à respecter les fonctions des interfaces prédéfinies.

8.2. CONVENTION CONCERNANT LES SCHÉMAS EXPLICATIFS

- Les différents blocs représentés par des rectangles sont les couches de l'application.

- La partie "Client" désigne le poste utilisateur qui demandera des services aux serveurs:

- La partie "Serveur" désigne un des serveurs qui fournit des services:

Le réseau est utilisé pour les "appels distants", c'est-à-dire lorsque deux modules ne se trouvent pas physiquement sur la même machine:

- Les sens des interfaces de communication entre les composants.

8.3. ARCHITECTURE N-TIER À 5 COUCHES

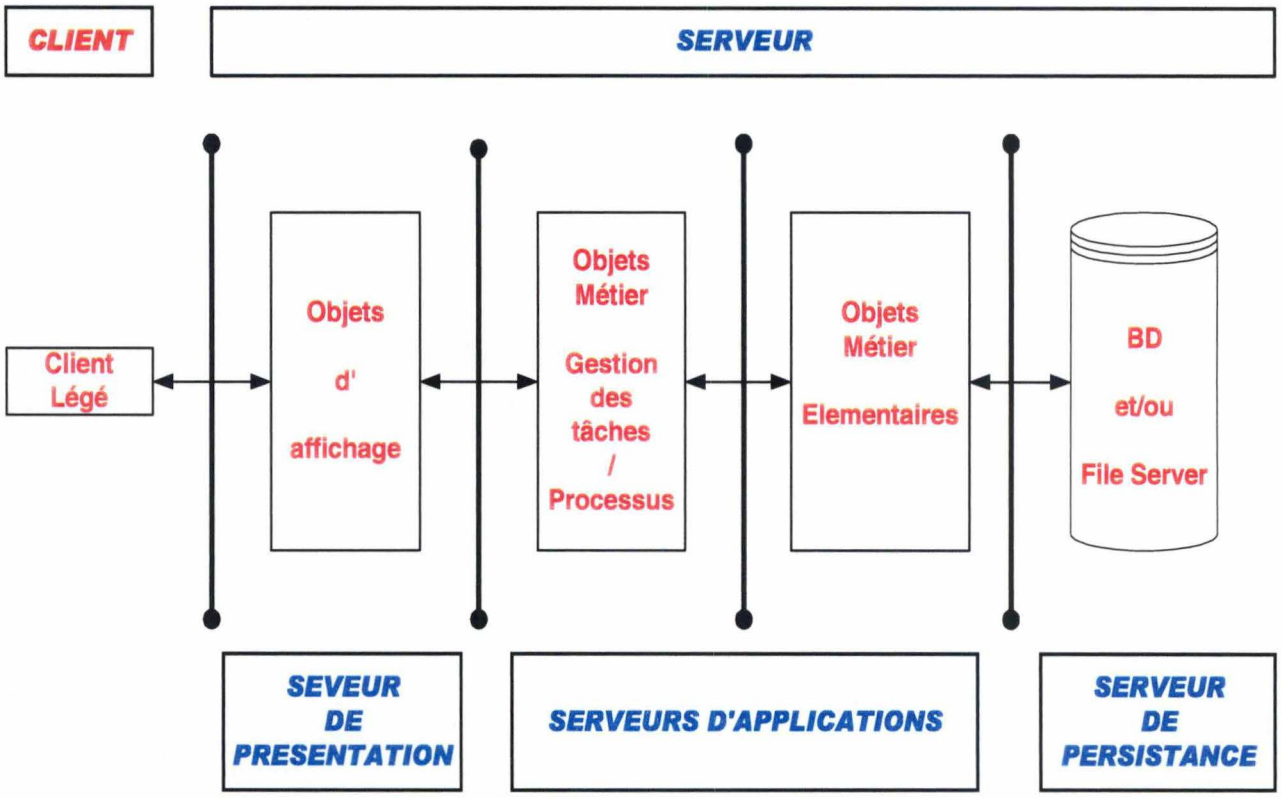


Schéma 4

Avantages/Inconvénients de l'architecture par couche

Avantages

- Elle imite, en terme de performance et en terme d'installation, la charge sur le poste client. En effet, à l'exception de la gestion de l'interface, tout est exécuté sur les différents serveurs. Cela facilite le load balancing.
- Elle limite à sa plus simple expression le déploiement sur le poste client.
- Elle restreint les problèmes d'installation à un nombre limité de serveurs.

- Elle augmente le niveau de sécurité et de confidentialité, en limitant les possibilités de reverse engineering et en permettant de mettre en place des niveaux de sécurité plus élevés sur un nombre limité de serveur.
- Elle favorise l'utilisation d'interface normalisée. Celles-ci sont développées, soit au niveau de l'équipe ou de la société, soit en se basant sur des protocoles standards disponibles sur le marché.
- Elle permet de paralléliser le développement.
- Elle permet le remplacement d'un module sans impacter les autres. Il suffit en effet de respecter les spécifications des interfaces inter-modules. Cela est vrai quel que soit le niveau de découpe dans un module. En effet la notion de découpe en modules doit être récursive dans un module, si la complexité du module le nécessite.
- Elle amène, à long terme et pour des environnements qui ne sont pas trop volatiles, une meilleure qualité de développement.

Inconvénients

- Elle augmente le niveau de complexité du déploiement et du tuning des modules utilisants des échanges d'informations à distance. En plus des problèmes propres à l'interface, on peut rencontrer des problèmes dus au réseau, à l'installation, aux différents niveaux de versions.
- Elle nécessite plus de compétence au niveau du déploiement des différents modules.
- Elle rend difficile la garantie d'une qualité de service. Celle-ci est grandement liée à la qualité des interconnections entre les différents serveurs.
- Elle demande un plus haut niveau de compétence de la part des équipes de développement.

- Elle nécessite une très grande coordination des différentes équipes pour la maintenance des différentes interfaces.
- Elle nécessite de la part de l'analyste une compétence d'un bout à l'autre de la chaîne.

8.4. ARCHITECTURE GLOBALE D'AGORAWATCH

Après analyse de différents outils et techniques disponibles, il a été décidé que

- L'architecture générale de l'application AGORAWATCH reposera sur une architecture n-tier, compatible J2EE (Java 2 Platform Enterprise Edition).
- La gestion de documents structurés se fera en utilisant le langage de définition des documents XML (Extensible Markup Language).

8.4.1. L'ARCHITECTURE N-TIER, COMPATIBLE J2EE

Couches majeures qui composent l'architecture J2EE

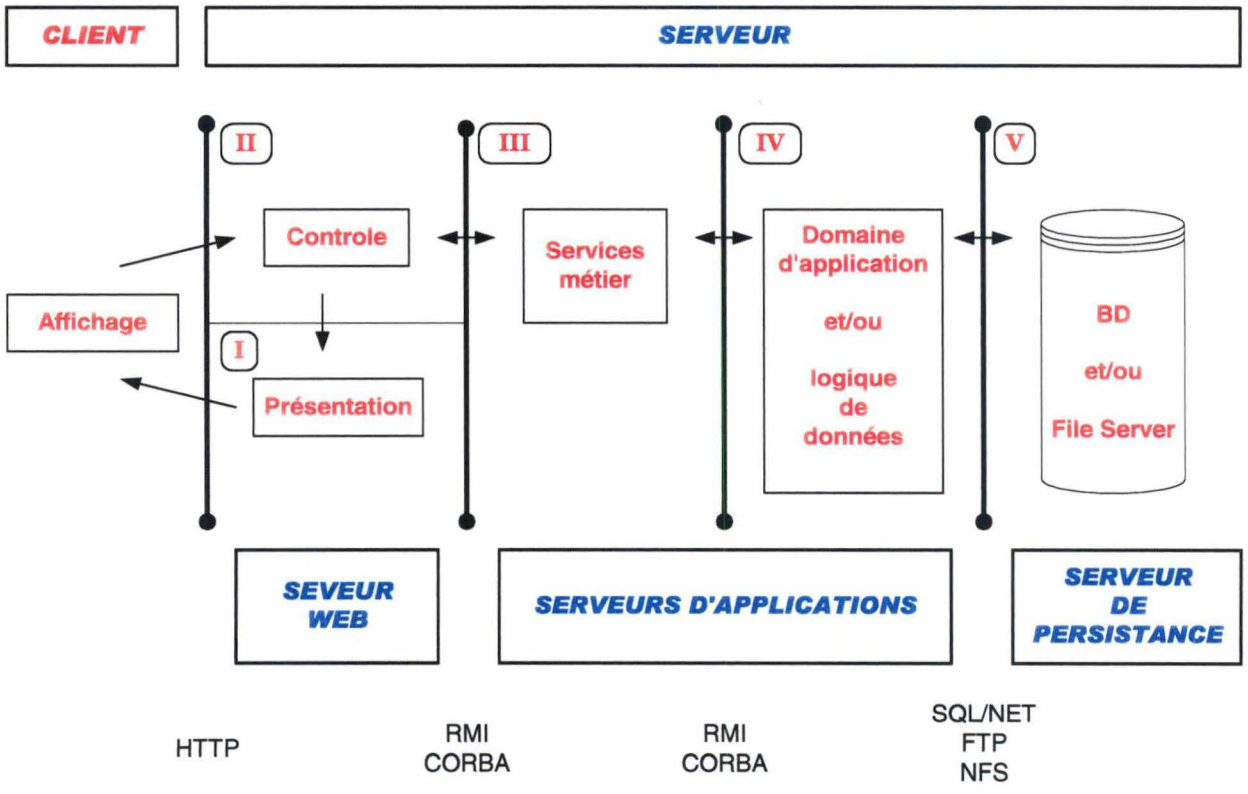


Schéma 5

I. La **couche de présentation** gère les interactions entrées-sorties avec l'utilisateur. Elle retournera du HTML au client. Elle présente les données de l'application et permet un contrôle rudimentaire des données entrées par l'utilisateur. Dans cette couche c'est le rôle joué par un composant **JSP** (Java Server Pages) ou le langage de transformation **XSLT** (eXtensible Style Sheet Transformation).

II. La **couche logique de contrôle** gère les interactions entre la couche de présentation et la couche métier. Intercepte les commandes émises par l'utilisateur et exécute les tâches métier associées. Elle cache la représentation distribuée du domaine de l'application à l'interface utilisateur. Et surtout elle maintient l'état de conversation pour la couche présentation. C'est le rôle des composants **servlet** de réaliser cette couche.

III. La **couche logique des services métier** fourni un API (Application Program Interface) vers les processus ou vers les tâches de l'application qui doivent être chaînées. C'est le rôle des composants **EJB Session** (Enterprise Java Bean) de réaliser cette couche.

IV. La **couche logique des données ou domaine de l'application** modélise les objets métiers élémentaires et elle est aussi responsable du mapping objet/relationnel. C'est le travail des **EJB Session** ou des **EJB Entité** dans cette couche.

V. La **couche persistance** gère les mécanismes nécessaires à la gestion de l'état des objets, en se reposant généralement sur une base de données relationnelle.

Intégration des composants J2EE dans les différentes couches.

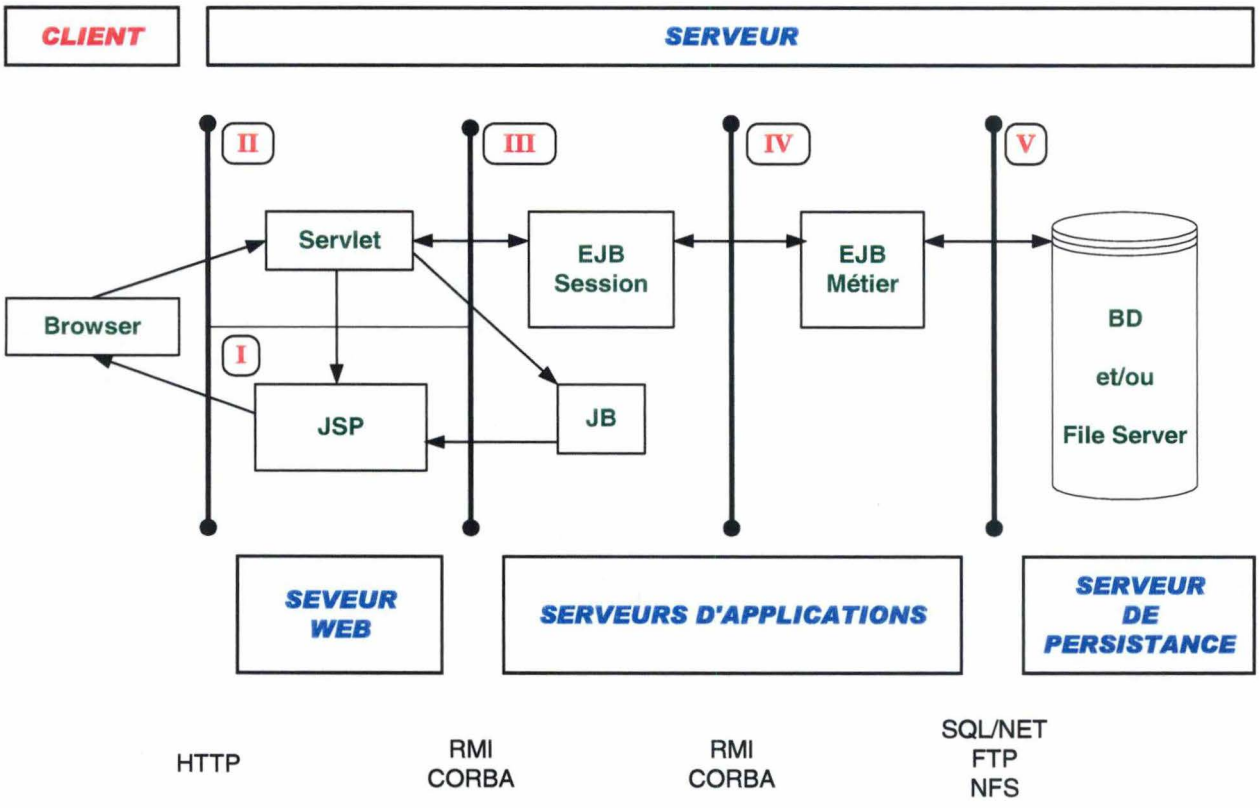


Schéma 6

Dans J2EE il y a cinq types de composants majeurs :

- Les **Java Bean** qui désignent la notion de composant dans le langage Java.
- Les **EJB Session** qui sont utilisés pour modéliser un processus ou des tâches de l'application
- Les **EJB Entité** qui modélisent le plus souvent des objets métiers du domaine de l'application.
- Les **Servlet** sont des classes Java qui peuvent aisément s'interfacer avec le protocole HTTP.
- Les **JSP** (Java Server Pages) sont des pages HTML auxquelles on a ajouté des comportements écrits en Java. Ces pages ne sont pas

interprétées par le client, mais compilées en servlet et exécutées par le serveur. Ce sont des composants de présentation typique.

Même si un servlet est capable de générer du HTML, il est plus facile d'un point de vue maintenance, de laisser gérer le comportement par un servlet et de laisser la partie présentation se faire par les JSP.

Il existe deux manières d'écrire des EJB Entité et des EJB Session, avec ou sans état. Les deux méthodes ont des avantages et des inconvénients. Elles répondent à des impératifs différents.

Les EJB avec état

Les **EJB avec état**, est une méthode facile pour gérer l'état conversationnel de l'application mais nécessite une instance par client, ce qui, si l'on a des dizaines de milliers de connexions simultanées, peut demander énormément de ressource CPU et mémoire. Une solution pour limiter la quantité de ressources est de donner à l'EJB un cycle de vie complexe, c'est à dire de permettre un état de passivation et d'activation. Quand L'EJB est rendu passif, on doit sauver la valeur des propriétés dans une base de données. Quand l'EJB est rendu actif, on doit récupérer les valeurs des propriétés dans la base de données.

Les **EJB avec état** sont pratiques pour des applications qui seront accédées par un nombre limité d'utilisateurs simultanés.

Les EJB sans état

Les **EJB sans état**, sont plus complexes à programmer. C'est à dire que l'entièreté des valeurs nécessaires à l'utilisation d'un **EJB sans état** doit pouvoir être sauvegardé dans une autre partie de l'application. Leur grand avantage est que l'on peut créer un pool de Bean. Quand un client a besoin d'un Bean il en fait la demande, obtient une référence, l'utilise pour l'invocation de ses méthodes et la rend au pool.

Les **EJB sans état** sont pratiques car ils permettent un grand nombre de clients simultanément, tout en ne consommant pas trop de ressource CPU et mémoire.

Interoperabilité et portabilité

J2EE assure une interoperabilité à deux niveaux :

- au niveau des différentes plates-formes du marché supportant java
- au niveau du monde non-java, grâce au support IIOP (Internet Inter-ORB Protocol) que java offre au serveur EJB

Exemple d'éxécution d'une requête à travers les différents composants d'une application J2EE.

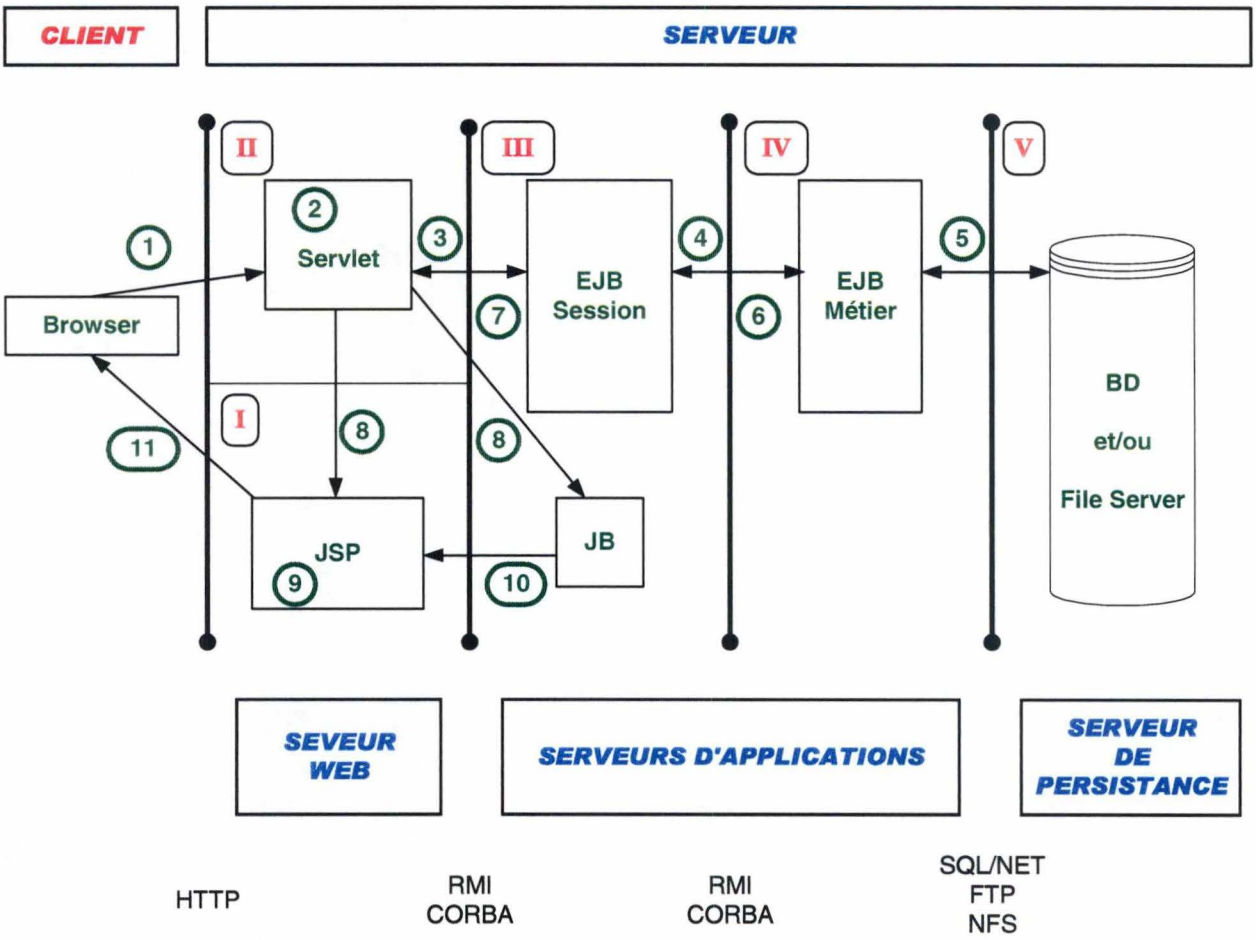


Schéma 7

A titre didactique, nous utiliserons l'affichage d'annuaire pour décrire l'enchaînement des composants utilisés dans le cadre d'une architecture J2EE.

Les composants d'une application J2EE sont indiqués en « **bold** »,

Les noms d'instance et de méthode sont indiqués en « *italic.* »

1. L'utilisateur clique sur l'hyperlien « Lire Annuaire ». Le navigateur crée une **URL**, y attache les paramètres nécessaires et envoie le tout via le protocole **HTTP**.

2. La requête sera examinée dès qu'elle arrivera au serveur **WEB**. Le serveur l'identifie comme une demande pour un **servlet** et transmet cette demande et ses paramètres au moteur de **servlet** pour exécution.

3. Le **servlet** interprète les paramètres et invoque l'interface Home d'un **EJB**. Le **servlet** obtient un **EJB Session** de type *LireAnnuaire*. Et invoque la méthode appropriée *get* sur *lireAnnuaire*.

4. Pour répondre à la méthode *get*, L'**EJB Session** instancie l'**EJB Entité** *Annuaire*, affecte les valeurs aux propriétés et invoque une méthode pertinente sur cet objet (*GetAnnuaire*).

5. L'**EJB Entité** (*Annuaire)* interagit avec une base de données pour obtenir l'état de l'objet *Annuaire*. Les méthodes *TrouveFichier* et *LireFichier* permettent de lire le fichier qui contient l'annuaire.

6. L'**EJB Entite** (*Annuaire)* retourne les informations à l'**EJB Session** (*LireAnnuaire)*.

7. L'**EJB Session** traite l'information puis passe le résultat au **servlet**.

8. Le **servlet** interprète le résultat si nécessaire, crée un **JavaBean** dynamique qui sert de transport d'informations vers la couche de présentation. Après quoi, le **servlet** passe la requête à une page **JSP**.

9. Le moteur **JSP** trouve le **JSP** demandé, le compile si nécessaire et crée un thread dédiée à l'intérieur de la **JVM** (Java Virtual Machine).

10. Le **JSP** récupère les informations du **JavaBean** créées par le **servlet** dans le contexte de la requête et incorpore l'information dans le flux **HTML** qu'il produit.

11. Le **HTML** produit est propagé via **HTTP** au navigateur, qui affiche l'annuaire.

Scalabilité d'un système.

Les questions suivantes permettent d'évaluer la taille, la puissance et les techniques d'implémentation pour réaliser une application. En fonction de la réponse à ces questions, il faut décider du nombre de couches et du type de composants que l'on doit utiliser pour répondre au mieux à la charge générée par l'utilisation de l'application.

- Combien d'utilisateurs simultanés doit-il supporter ?
- Peut-on se contenter d'utiliser des EJB avec état ?
- Doit-on utiliser un pool de connexions à la base de données ?
- Comment manipuler les grands volumes de données retournées par des requêtes ?

8.4.2. COMPOSANTS DE LA NORME XML

XML (Extensible Markup Language) est le résultat de la coopération d'un grand nombre d'entreprises et de chercheurs partenaires du World Wide Web Consortium (W3C). L'objectif principal du consortium était de définir un formalisme simple permettant l'échange de documents complexes sur le WEB.

XML est un langage de description et d'échange de documents structurés.

XML est un sous-ensemble de SGML, qui possède les mêmes objectifs que SGML, le balisage de tout type de données. Mais il a été

débarrassé de toute la complexité superflue pour une utilisation sur le WEB.

Comme XML est un subset de SGML, tout document qui répond à la syntaxe XML est compatible SGML. L'inverse n'est pas forcément vrai.

Par contre, il est important de comprendre que XML est plus qu'un « langage », c'est un standard de création de langages respectant les critères XML.

Structure d'un document XML

Un document XML se compose :

- D'une en-tête, dont la présence est facultative mais fortement conseillée.
- D'un arbre d'éléments. Il forme le contenu proprement dit du document.
- De commentaires et d'instructions de traitement, dont la présence est facultative, et qui peuvent apparaître aussi bien dans l'en-tête que dans l'arbre des éléments.

Un document qui obéit aux règles syntaxiques du langage XML, est un document XML « bien formé ».

L'en-tête sert à déclarer la version du langage XML, le type de code caractères utilisés dans le document, et des déclarations concernant des documents extérieurs qui doivent être pris en compte pour le traitement.

ex : <?xml version='1.0' encoding='ISO-8859-1' ?>

< ! - - ceci est un commentaire - - >

Les éléments de l'arbre d'un document XML se composent d'une balise d'ouverture, d'un contenu d'élément, et d'une balise de fermeture.

ex : <nom>Defreine</nom>

- <nom> : balise d'ouverture
- Defreine : du contenu de l'élément
- </nom> : balise de fermeture

La balise d'ouverture peut en plus contenir des attributs. Un attribut sert à décrire une propriété de l'élément. Un attribut a la forme nom='valeur'

ex : <membre niv_secu='1'> ... </membre>

- niv_secu : nom de l'attribut
- '1' : valeur de l'attribut

Exemple d'un document XML bien formé

Annuaire.xml est un fichier XML bien formé Il contient les données d'un annuaire.

```
<?xml version='1.0' ?>

<annuaire owner='DBA' >
     <membre niv_secu='1'>
          <nom>Defreine</nom>
          <prenom>Pierre Jean</prenom>
          <num_membre>1234</num_membre>
     </membre>
     <membre niv_secu='0'>
          <nom>Brutto</nom>
          <prenom>Calogero</prenom>
          <num_membre>4567</membre>
     <membre>
</annuaire>
```

Utilisation de langage XML dans le cadre d'AGORAWATCH

Comme XML permet de définir un langage, il suffit donc dans le cadre d'AGORAWATCH de d'écrire le langage qui va répondre aux différentes contraintes fonctionnelles du domaine de l'application.

Pour cela, on va définir à l'aide des balises et des attributs, des éléments métier, des attributs de sécurité, des attributs de chiffrement et tous les éléments/attributs nécessaires au domaine de l'application. Puis nous associerons ces balises et attributs aux différents traitements de l'application.

Pour réaliser la définition des balises, des attributs et des éléments, on va utiliser une autre partie de la norme XML, le schéma.

Les schémas XML

Un schéma XML est avant tout un document XML bien formé, pour lequel un certain nombre d'éléments, d'attributs et de traitements ont été définis.

De plus le schéma permet de typer les éléments qui composent le langage.

Exemple de fichier schéma

Annuaire.xsd fichier XML bien formé, contenant le schéma de définition de la structure du document annuaire.xml

```
<?xml version='1.0' ?>
<xs:schema      xmlns='http://www.w3.org/1999/XMLSchema'
          xmlns:xs=' 'http://www.w3.org/1999/XMLSchema' />

<xs:annotation>
     <xs:documentation> Exemple de schéma définiton d'un
     annuaire
     </xs:documentation>
</xs:annotation>

<xs:complexType name='annuaire'>
     <xs:attribute name='owner' type='xs:string'
           use='required' />
     <xs:element name='annuaire' type='membre' minOccurs=1
           maxOccurs='unbounded' />
          </xs:complextype>

<xs:complexType name='annuaire'>
     <xs:attribute name='niv_secu'
               type='xs :non-negative-integer'
               use='required' />
     <xs:element name='nom' type='xs :string'
                    minOccur='1' maxOccur='1'/>
     <xs:element name='prenom' type='xs :string'
                    minOccur='1' maxOccur='1' />
     <xs:element name='num_membre'
                    type='non-negative-integer'
                    minOccur='1' maxOccur='1'/>
</xs:complexType>
```

Cet exemple montre que l'on peut :

- Définir des éléments de type simple ou complexe. Que les éléments complexes peuvent contenir des éléments simples ou complexes.
- Définir des cardinalités minimum et maximum pour un élément.
- Définir des attributs pour un élément.

Le but de l'architecture n'est pas de présenter une liste exhaustive de l'entièreté du langage des schémas XML. Pour les personnes qui sont intéressées une description complète du langage est disponible sur le site http://www.w3c.org/XML/Schema .

Grâce au schéma XML, on va pouvoir définir des structures de document XML.

Un document XML bien formé qui obéit à une structure définie, est un document XML « valide ».

Dans le cadre de notre application les structures de document XML seront appelées **template**.

Pour le moment, nous avons montré que nous pouvons gérer des informations au format XML, et que ces informations répondront aux contraintes fonctionnelles nécessaires à l'application. Il faut maintenant être capable d'exploiter ces données dans les applications que nous désirons développer. Pour cela, il faut pouvoir lire les données mais aussi les valider, les manipuler, les modifier, les créer, les supprimer. Il est donc nécessaire d'avoir à notre disposition des interfaces avec notre code application.

Actuellement, il existe deux grandes méthodes de manipulation des informations au format XML, DOM (document object model) et SAX (Simple API for XML).

DOM

DOM est un modèle objet de document. Cette méthode permet de manipuler un document dans sa globalité, d'accéder en ligne directe à une partie de la structure, d'ajouter, de modifier et de remplacer des éléments dans le document original.

En d'autres termes, DOM est un API qui permet de charger en mémoire une image de la structure d'un document et des éléments qui le composent.

DOM est très performant quand on désire accéder et manipuler des sous-ensembles de la structure d'un document.

Par contre, c'est une méthode qui nécessite des ressources mémoires et qui peut se montrer relativement lente pour le traitement de document volumineux.

SAX

Au contraire de la méthode DOM, SAX a été développé pour manipuler les documents volumineux. SAX est un parseur qui lit le flot de données XML en entrée, reconnaît et interprète les marques de balisage au fur et à mesure qu'il les rencontre. Chaque balisage reconnu est immédiatement passé à l'application.

C'est un modèle de traitement dirigé par les événements.

SAX a les inconvénients des ses avantages :

- SAX ne permet pas de contrôler l'ordre de recherche du parseur.
- SAX est en lecture seule.

Couche de transformation

Avant de décider de programmer des applications spécifiques qui géreront les balises spécifiquement décrites pour répondre aux contraintes fonctionnelles du domaine de l'application. Il est possible d'envisager l'utilisation d'un langage de transformation mis à disposition dans la norme XML.

Ce langage de transformation est XSLT (eXtensible StyleSheet Langage Transformations).

Indépendamment de tout formatage, XSLT permet de traiter un arbre de données pour produire un autre arbre de données. XSLT est un langage de programmation, que l'on peut classer dans les langages de type fonctionnels déclaratifs.

En utilisant les XSLT on règle aussi le problème de la mise en page. En effet, afficher un arbre de données consiste principalement à faire subir une transformation aux données pour lui ajouter des tags d'un langage d'affichage. On utilisera donc le langage XSLT pour transformer un arbre de données XML en XHTML (eXtensible HyperText Markup Language) ou tout autre type de mise en pages.

Exemple de fichier XSLT

Annuaire.xsl est fichier XML bien formé contenant un stylesheet d'affichage du fichier de données annuaire.xml.

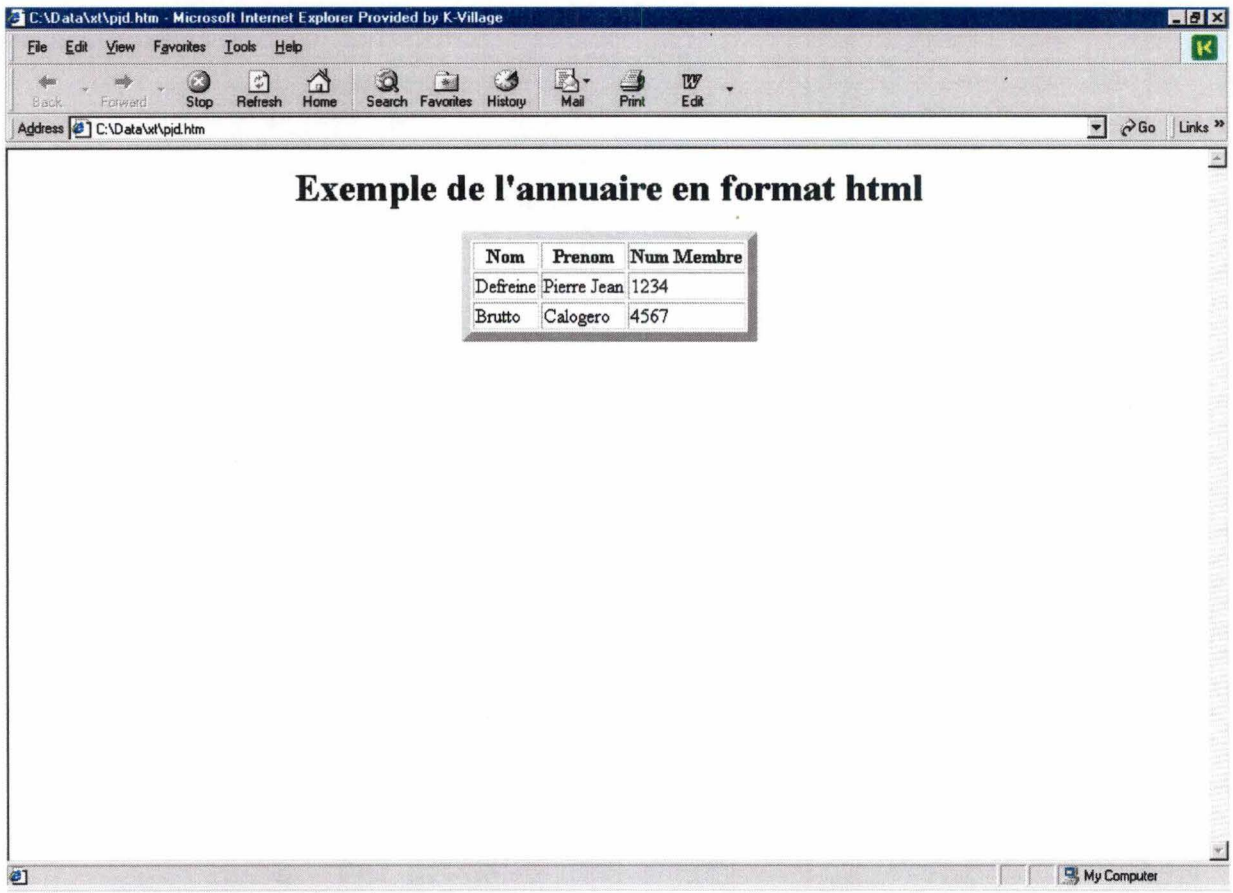
```
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/TR/REC-html40"
                version= '1.0' >

<xsl:template match="/" >
<html>
   <head><h1><center> Exemple de l'annuaire en forma
thtml</center></h1>
   </head>
   <body><table border="8" align="center" >
      <tr><th>Nom</th><th>Prenom</th><th>Num
Membre</th></tr>
      <xsl:for-each select="annuaire/membre"><tr>
         <td><xsl:apply-templates select="nom" /></td>
         <td><xsl:apply-templates select="prenom" /></td>
         <td><xsl:apply-templates select="num_membre"
/></td>
      </tr></xsl:for-each>
   </table></body>
```

```
</html>
</xsl:template>
</xsl:stylesheet>
```

résutat du parsing du fichier annuaire.xml sur lequel on a appliqué la stylesheet annuaire.xsl.

```
<?xml version="1.0" encoding="utf-8"?>
<html xmlns="http://www.w3.org/TR/REC-html40">
<head><h1><center> Exemple de l'annuaire en format html
</center></h1></head>
<body>
<table border="8" align="center">
<tr><th>Nom</th><th>Prenom</th><th>Num Membre</th></tr>
<tr><td>Defreine</td><td>Pierre Jean</td><td>1234</td></tr>
<tr><td>Brutto</td><td>Calogero</td><td>4567</td></tr>
</table>
</body>
</html>
```

Visualisation du code HTML généré dans une browser.



En résumé

XML et les différentes normes associées qu'il propose permet de gérer d'une façon simple, intégrée, standardisée et évolutive des documents structurés.

8.4.3. CONCLUSION

L'utilisation d'une architecture en couche compatible J2EE permettra par sa modularité de garantir une très bonne évolutivité de

l'application en fonction des impératifs de performance, de sécurité et de standard qui seront rencontrés tout au long de la vie de l'application.

Le XML, lui, permet de rencontrer l'objectif majeur que doit réaliser l'application et de gérer des documents dont les données sont structurées.

8.4.4. CHOIX DES COMPOSANTS TECHNIQUES EN FONCTION DES DIFFÉRENTES COUCHES

Le **couche client** utilisera un Navigateur compatible html 4, Java script et applet

La **couche de présentation** fournira des objets d'affichage à base de JSP (Java Sever Pages) ou de servlets qui manipuleront du XML et du XSLT.

La **couche logique de contrôle** fournira des objets de contrôle à base de servlet et ne devra en aucun cas fournir des objets d'affichage.

.La **couche logique des services métier** fournira des objets métiers EJB Session qui géreront les process et les tâches de l'application.

La **couche logique des données** fournira des objets métiers EJB Métier, les composants métiers. Ils utiliseront entre autre des classes de gestion de document à base de DOM ou SAX.

La **couche persistance** qui à base de fonctions de gestions de fichier plat. Il n'est pas prévu, dans un premier temps, et si les performances sont suffisantes, de stocker les informations dans une base de données de type relationnelle. Quoi qu'il en soit, les méthodes d'accès aux données devront être écrites de façon à permettre l'intégration d'un système de base de données en ne nécessitant pas d'impacts significatifs sur les autres méthodes.

8.5. ARCHITECTURE GLOBALE D'AGORAGET

L'architecture générale de l'application AGORAGET reposera sur une architecture n-tier fat (gros) client, light (légé) serveur en étoile, la partie principale se trouvant sur le client.

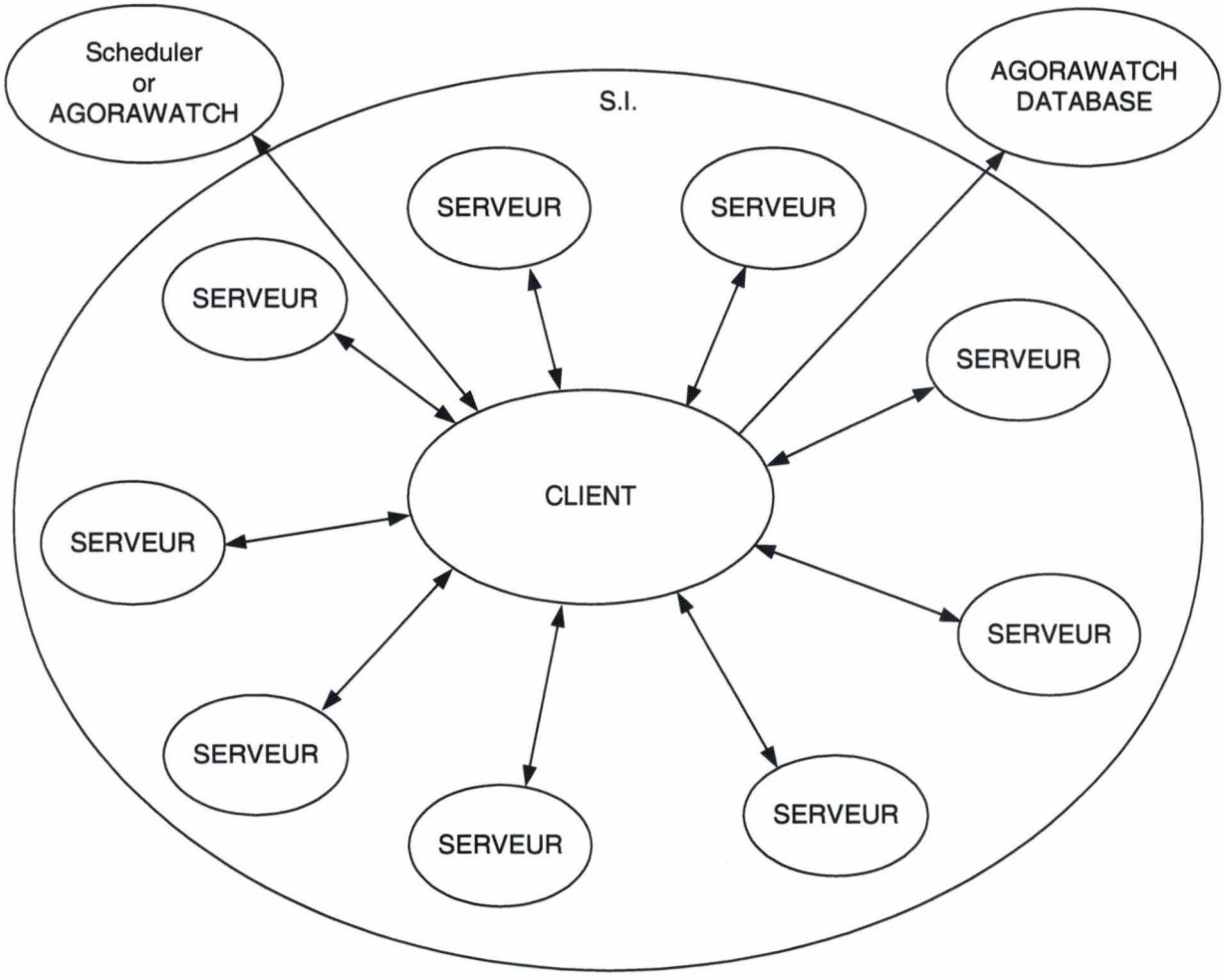


Schéma 8

Le scheduler ou AGORAGET interagit avec le client pour demander l'exécution de récoltes d'information. Le client joue le rôle de chef d'orchestre et demande à tous les serveurs l'information souhaitée. En dernier lieu, les informations récoltées sont transmises à AGORAWATCH.

8.5.1. DESCRIPTION LOGIQUE DES COUCHES DE L'APPLICATION AGORAGET

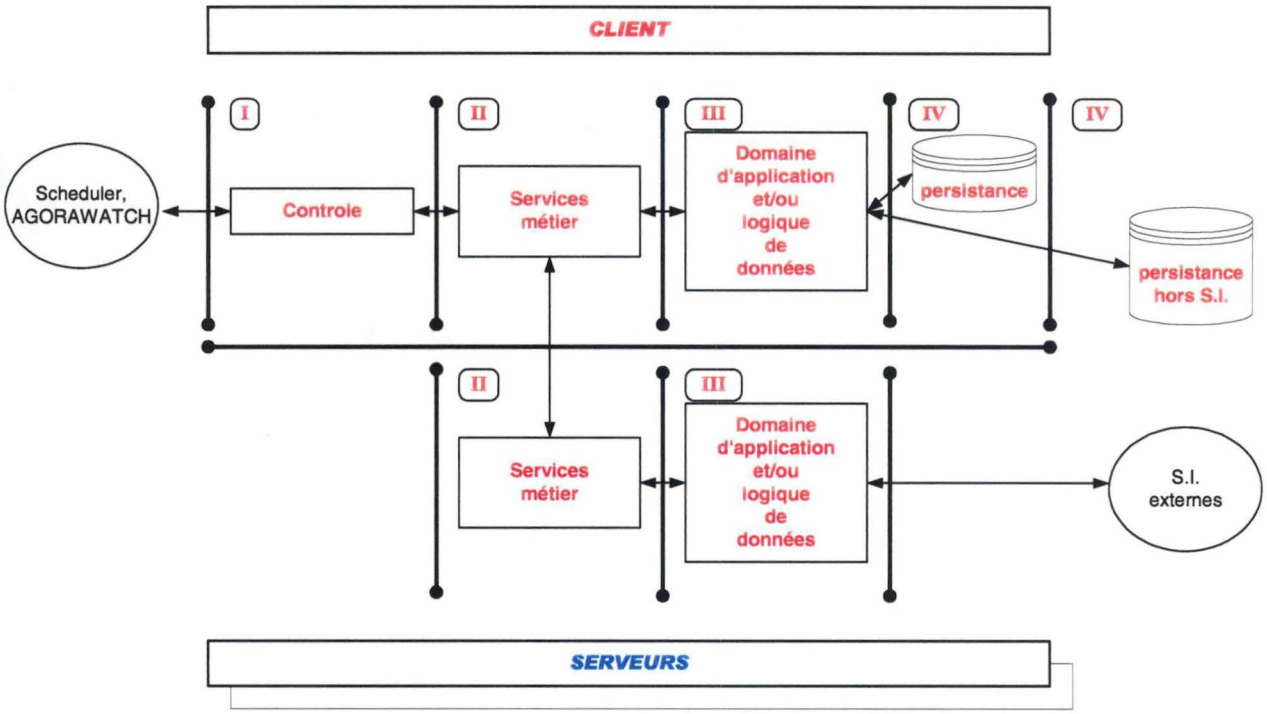


Schéma 9

La couche logique de contrôle (I)

Cette couche reçoit d'AGORAGET ou du scheduler les commandes à exécuter. Elle contrôle la validité de celles-ci et demande l'exécution des tâches métier associées à la couche logique des services et, ensuite, retourne uniquement le statut de l'exécution de la commande demandée.

La couche logique des services métiers (II)

Cette couche réalise le processus lié au service demandé. Cette réalisation passe par une série d'échanges avec la couche logique des données. Cette couche est découpée en deux parties. La première partie

est localisée sur le client et contient les services métier qui fournissent les services destinés à la gestion du S.I. lui-même. Ces services communiquent avec la couche logique de données client pour accéder aux objets de base. La deuxième partie est localisée sur les serveurs externes et contient les services métier destinés à la récolte d'information sur ces mêmes S.I. externes. Elle communique avec ces mêmes S.I. via la couche logique de données serveur.

La couche logique des données (III)

Cette couche modélise les objets métier élémentaires et est aussi responsable de la communication avec la couche persistance et AGORAWATCH (sur le client) ou avec les S.I. externes (sur les serveurs).

La couche persistance (IV)

Cette couche gère les mécanismes nécessaires à la sauvegarde des informations du S.I. dans une base de données ou à l'aide de systèmes de fichiers. Les informations provenant de la récolte sur les S.I. externes est stockée dans AGORAWATCH.

8.5.2. DESCRIPTION DES CHOIX TECHNIQUES

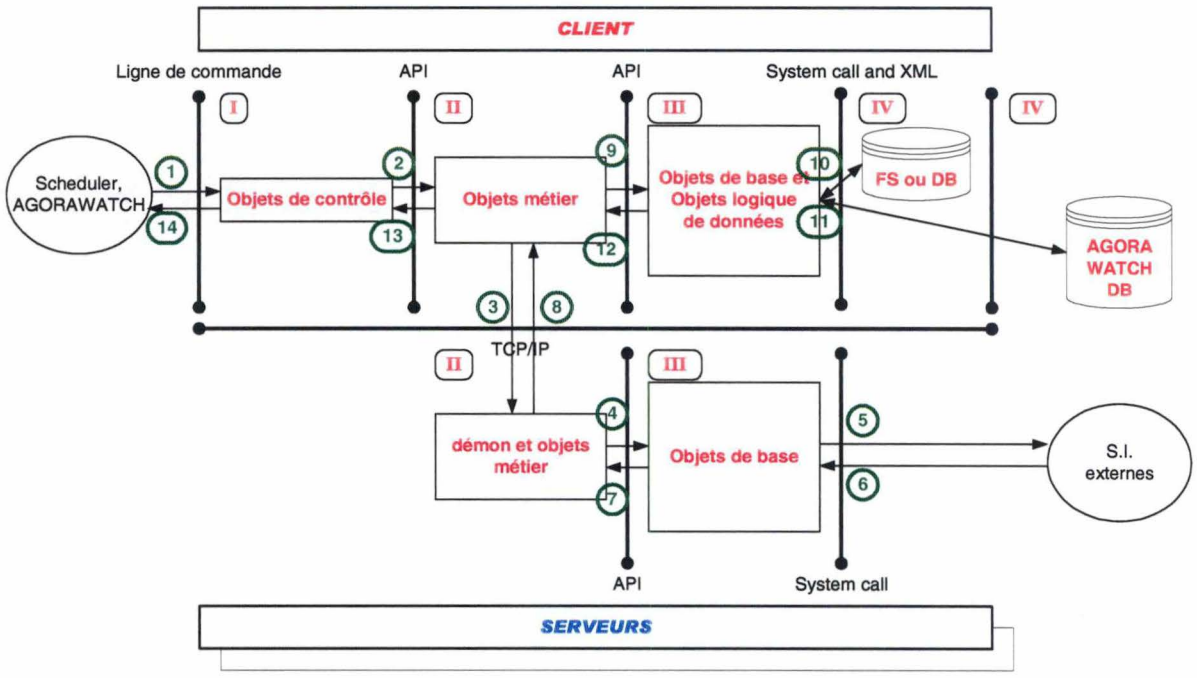


Schéma 10

Choix techniques:

1. La partie client sera localisée sur un serveur SUN.
2. Le langage de développement pour les différentes couches **CLIENT** sera JAVA.
3. Le parseur utilisé pour générer le XML à destination d'agorawatch sera TurboXML de WebGain.
4. La partie serveur sera localisée sur les S.I. externes.
5. Le langage de développement pour les différentes couches **SERVEUR** sera le "C".

Raisons

1. La plate-forme SUN est privilégiée chez Mobistar pour tous les futur développements. Elle possède une parfaite intégration avec le standard J2EE.

2. Le langage JAVA est Langage privilégié chez Mobistar. Il est incontournable dans le framework J2EE. Il est portable. Il permet une parfaite intégration avec XML et ses parseurs.

3. Mobistar utilise le framework J2EE "WebLogic Application Server" (leader sur le marché) ainsi que l'environnement de développement WebGain (de WebLogic), le parseur XML intégré de WebGain s'impose donc.

4. La récolte d'informations devant se faire sur les systèmes externes, une partie du S.I. doit se trouver sur chaque S.I. externe. Cette répartition est le but de notre S.I: pouvoir exécuter des commandes telles "df -k" sur UNIX pour obtenir la liste des file system ainsi que leur taux de remplissage...

5. L'écriture de démons portables sur tous les systèmes opératoires UNIX (SUN, HP, DIGITAL) ainsi que sur WINDOWS NT ne peut se faire que via un langage portable et réputé sur chacune de ces plates-formes. De plus, ce langage doit être compilable pour générer des exécutables. Ceci pour éviter un reverse engineering possible avec certains langages tel java et le byte code qu'il génère. Le langage "C" s'impose donc.

8.5.3. DESCRIPTION PHYSIQUE DES COUCHES DE L'APPLICATION AGORAGET

La couche objets de contrôle (I)

La couche objets de contrôle reçoit des commandes provenant du scheduler ou de AGORAWATCH et communique avec l'application AGORAGET via un appel à un exécutable. Les paramètres de cet exécutable permettent de qualifier la récolte d'information à effectuer. Le choix du simple exécutable permet d'interfacer simplement AGORAGET, AGORAWATCH et le scheduler Mobistar (Unicenter). Le scheduler peut en effet exécuter une commande sur n'importe quel serveur de Mobistar et donc, il sera capable d'appeler AGORAGET sur le client. AGORAWATCH se trouvera sur la même machine qu'AGORAGET. Si cela n'était pas le cas, AGORAWATCH pourait appeler l'exécutable via les commandes UNIX standard d'appel à distance: Les "remote commands". Si la sécurité des "remote commands" n'est pas satisfaisante, les appels à distance utiliseront alors SSH.

Les autres couches (II, III, IV)

Sur le client:

Ces couches communiqueront entre elles par l'intermédiaire d'interfaces (API). Ces interfaces seront des classes java. Chaque couche pourra être programmée par des équipes différentes. Seul l'API sera le point de contact entre les couches et donc entre les équipes. La définition de ces API sera la première chose à réaliser lors de la prochaine étape. La couche DB sera en fait implémentée par une série de fichiers plats XML. Il n'y aura donc pas de base de données au sens relationnel du terme. La communication entre la couche objets de base et la base de données AGORAWATCH se fera à travers un échange de fichiers XML quel que

soit le stockage qu'utilise AGORAWATCH, ceci pour garantir l'indépendance de ces deux sous applications.

Sur le serveur:

Nous trouverons ces couches qui communiqueront entre elles également par l'intermédiaire d'interfaces (API). Ces interfaces seront des librairies de fonctions écrites en "C" étant donné le choix réalisé plus haut.

Entre les deux:

La communication entre le client et les serveurs se fera en utilisant TCP/IP. Les serveurs hébergeront chacun un démon que nous appellerons "AGORAGET**D**". Ce démon restera constamment à l'écoute sur une porte qui sera à définir avec l'équipe sécurité. A chaque demande du client, une connexion TCP/IP sera établie jusqu'à la fin de la transaction.

8.5.4. EXEMPLE DE COMMUNICATION ENTRE LES DIFFERENTES COUCHES.

1. Le scheduler ou AGORAWATCH envoie une commande à la couche logique de contrôle via un appel à un exécutable et des paramètres. Elle demande par exemple via les paramètres adéquats de récupérer la taille et l'espace libre des systèmes de fichier du serveur UNIX "α".

2. La couche logique de contrôle, après avoir vérifié la validité de la commande reçue, demande à la couche objets métier la réalisation du service correspondant à la commande.

3. La couche objets métier du client demande à la couche objets métier du serveur UNIX "α" la réalisation de la commande reçue. Cette demande est réalisée par le client en se connectant au démon AGORAGETD sur le serveur "α". La couche objets métier du serveur

décompose la commande en commandes de base et demande la réalisation de ces commandes à la couche objets de base du serveur. Il n'y aura ici qu'une seule commande de base.

4. La couche objets de base du serveur traduit alors cette (ou ces dans d'autres cas) commande(s) de base en commandes compréhensibles par le système d'exploitation sur lequel il se trouve. Dans notre cas, la commande de base sera transformée en "df -k" (ce qui demande à un système UNIX la liste et l'occupation des systèmes de fichiers qui le composent).

5. Le "df -k" est envoyé au serveur UNIX "α" pour exécution via un appel système.

6. Le résultat de la commande est renvoyé à la couche objet de base. Par exemple:

```
Filesystem            kbytes    used   avail capacity  Mounted on
/proc                      0       0       0     0%    /proc
/dev/md/dsk/d1       1527116  937192  528840    64%    /
fd                         0       0       0     0%    /dev/fd
/dev/md/dsk/d5       2057501  884155 1111621    45%    /var
/dev/md/dsk/d4       1614299  140067 1425804     9%    /opt
/dev/md/dsk/d6       1018382  277531  679749    29%    /tmp
/dev/md/dsk/d7        324271  235940   55904    81%    /schedulertool
/dev/md/dsk/d20      6520109 2395965 4058943    38%    /app
/dev/md/dsk/d23      2155566   39234 2073221     2%    /prod
/dev/md/dsk/d0      41297032 32669709 8214353    80%    /DATA
/dev/md/dsk/d8         1069902   683293   333114      68%      /app/idm
```

7. La couche objet de base envoie l'information à la couche objets métier sur le serveur.

8. La couche objets métier sur le serveur envoie l'information sur la couche objets métier sur le client.

9. La couche objets métier sur le client transmet les informations reçues à la couche objets de base pour écriture des résultats dans AGORAWATCH.

10. La couche objets de base stocke les informations relatives à la commande reçue (logging de la demande...) dans le système de fichier du client.

11. La couche objets de base convertit les informations reçues en fichier XML compréhensible par AGORAWATCH.

9. Conclusion

Dans ce travail, nous avons étudié le problème de la gestion de l'information au sein d'une communauté de personnes à partir de l'étude d'un cas réel d'un département informatique de production.

Le problème de la gestion de l'information nécessaire au bon fonctionnement d'un département informatique est rarement abordé. Il semble impensable qu'un département informatique qui met à disposition des utilisateurs toute une série de systèmes d'information correspondant aux besoins spécifiques de chacun de ces départements ne possède pas lui-même un système d'information pour gérer sa propre information. Et pourtant c'est généralement le cas. Les informaticiens préfèrent généralement utiliser de petites solutions rapides répondant chacunes à leurs besoins propres plutôt que d'imaginer une solution globale. De plus, s'il est communément admis qu'il est nécessaire d'utiliser un système d'information pour gérer l'information des clients, des fournisseurs, du stock..., il n'en va pas de même pour l'information de ceux qui gèrent les systèmes d'information. Ce n'est pas un besoin 'business" direct et donc il est souvent difficile de faire passer l'idée ainsi que d'avoir du temps et du budget à y consacrer. Il est amusant de remarquer que s'il existe toute une série d'applications de gestion clientèle, fournisseur... répondant à tous les besoins business possibles et imaginables, il est très difficile d'en trouver ne fusse qu'une dont le but est de gérer un département informatique.

Notre recherche s'est alors portée sur un outil de gestion de l'information structurée au sens large c'est à dire pouvant gérer tout type d'information structurée et pouvant intégrer tout nouveau type d'information structurée suivant les besoins des utilisateurs. Nous étions donc plus à la recherche d'un logiciel générique que d'une application spécifique.

La plupart des outils rencontrés ne répondent pas à la notion d'information structurée, leur niveau de granularité le plus bas étant

généralement le document. Lors de cette recherche, nous nous sommes rendu compte que la notion de structure de l'information commence à percer avec l'émergence de standards basés sur XML (eXtensible Markup Language). L'intégration d'XML n'en est encore qu'a ses débuts car XML est encore trop souvent utilisé uniquement comme format d'échange (EDI) et non comme format de stockage bien qu'il en soit tout à fait capable.

N'ayant pu trouver d'outils capables de répondre à nos besoins, nous nous sommes penchés sur une solution orientée développement. Ayant étudié XML lors de notre recherche d'outils, ce standard nous est apparu comme étant la base de la solution que nous devons fournir. XML étant un dérivé de SGML, il apparaît comme étant parfait pour la gestion de documents ainsi que d'information structurée. XML et certains standards dérivés sont maintenant intégrés dans la plupart des outils disponibles sur le marché en tant que format d'échange. XML permet en effet à la fois le formatage de documents à des fins d'échange entre applications, la communication de messages entre des systèmes et même l'enregistrement de données dans une banque de données. De plus, un tel document message ou fichier contient encore en lui même des informations sur la structuration des uns et des autres, de sorte qu'un autre système puisse, la cas échéant, interpréter l'information tout à fait automatiquement.

Nous pouvons conclure que l'étude de l'intégration d'XML dans le cadre de la gestion de l'information au sein du département informatique nous a ouvert de grande perspectives. En effet, la non gestion de toute cette information conduit à une perte de temps, une perte d'information, un service à nos clients internes de moins bonne qualité et donc une perte d'efficacité par rapport à la concurrence. L'intégration de toute cette information éparse, qu'elle soit collectée automatiquement ou manuellement, permettra aussi bien aux employés, aux managers qu'au

directeur du département d'avoir accès directement à une information précise et à jour.

10.Références bibliographiques

F. Bodart, Y. Pigneur, *conception assistée des systèmes d'information*, Masson, Paris, France, 1994

F. Berqué, S. Frezefond, L. Sorriaux, *Java-XML et Oracle*, Eyrolles, Paris, France, 2001

J.L. Hainaut, *Ingénierie des bases de données*, Facultés universitaires notre-dame de la paix, Namur, Belgique, 1999

D. Hunter, C. Cagle, D. Gibbons, N. Ozu, J. Pinnock, P. Spence, *Initiation à XML*, Eyrolles, Paris, France, 2001

N. Kettani, D. Mignet, P. Paré, C. Rosenthal-sabroux, *De Merise à UML*, Eyrolles, Paris, France, 1998

G. Kindermans, *XML colle universelle*, IT press, p28-30

C. Larman, *Applying UML and patterns*, Prentice hall, New Jersey, USA, 1997

A. Michard, *XML language et applications*, Eyrolles, Paris, France, 2000

11.GLOSSAIRE

1595

Service de téléphonie fixe de Mobistar.

Administrateur

Dans le cadre de notre logiciel: Un administrateur est un utilisateur normal qui posséde (en plus des priviléges utilisateur) des privilèges spécifiques liés à la gestion globale du S.I.

API

Application Program Interface. Un API est un ensemble de routines, protocoles et outils pour aider au développement d'une application en fournissant les briques des briques de base.

ARS

The ARS tool is Mobstar's internal trouble ticketing system.

Batch

Se dit d'une application qui n'est pas interactive.

Billing

Nom usuel des applications concernant le domaine de la facturation.

Browser

Nom désignant une application permettant d'accéder généralempent à des serveurs web. Ces application servent généralement à surfer sur internet. Les plus connues sont Internet Explorer (Microsoft) et Netscape.

DASHBOARD

Tableau de bord: Ces tableaux sont la partie visible finale des KPI du département. Ce sont générallement des graphiques montrant les tendances des KPI.

DOCUMENTUM

Documentum est l'application de gestion documentaire utilisée dans certains départements de Mobistar.

DOD

Deployment and Operational Department: Ce département s'occupe du réseau GSM et GPRS, de son étude, sa mise en place, son évolution à sa gestion quotidienne.

DOM

Document Object Model. Mécanisme d'accès à un document XML. Ce mécanisme crée un arbre d'éléments avec des interfaces pour parcourir cet arbre. DOM est défini par une spécification du W3C.

DRP

Disaster Recovery Plan: Plan de récupération après désastre.

Duty report

L'équipe responsable des applications, des systèmes, des databases, ainsi que des réseaux sont de garde 24 heure sur 24. Il y a un roulement hebdomadaire de team member pour chaque équipe concernée. A la fin de chaque semaine de garde, le team member de chaque équipe réalise un rapport de garde: c'est le Duty report.

E.R.

Evolution Request: Les demandes de changement de fonctionnalités des applications sont regrouppées dans des ER. Une ER peut concerner plusieurs applications. Ces ER sont transmises à nos fournisseurs d'application (aucun développement n'est réalisé au sein de Mobistar) qui nous fournissent alors un prix et un delai de réalisation.

EJB

Enterprise Java Bean. Architecture de composants pour le développement et le déploiement d'applications d'entreprises distribuées orientées objet.

EIP

Enterprise information portal

Element

Un template est constitué d'une combinaison d'éléments. Un élément peut être un texte, un hyperlien interne, un hyperlien externe, un lien typé ou une zone.

GSM

Le "Global System for Mobile" est un des système pour téléphones digitaux cellulaires parmis les plus répandus.

Instance

Une instance d'information est un document ayant une structure se rapportant à une entité d'information (template). Exemple: Les informations concernant la base de donnée "DbClient" est une instance du type "Database" (qui définit la structure du document "database").

IOP

Informatique opérationelle. Nom du département qui est à l'origine du besoin du S.I. ici développé. Ce département est responsable de la mise en production des applications (fournies par ISD) et de leur suivi journalier.

ISD

Information System Department. ISD est le nom du département réalisant le développement informatique chez Mobistar.

ISP

Internet Service Provider. Un ISP fournit des services internet (connection au réseau internet, ébergement de site web, création de site web...).

JBD

Jaba Bean. Classe java susceptible d'être manipulée par un outil de développement visuel et assemblée pour former une application.

JVM

Java Virtual Machine. Une JVM est une application permettant d'exécuter du byte-code. Le byte code est généralement produit à partir de classe Java. On peut trouver des JVM pour les opérating system et les plateformes les plus utilisées actuellement. De ce fait, un même byte-code peut être exécuté sur des machines tout à fait différentes. C'est sur les JVM que se base la portabilité du code Java.

k-village

k-village est le nom ainsi que l'URL du site intranet de Mobistar. Ce site contient des informations générales sur les différents départements de Mobistar (Human ressource, sales, technique, informatique...).

KPI

Key performance indicator: Un KPI est généralement appliqué à une activité qui est importante pour le management. Il est constitué d'éléments mesurables qui permettent d'évaluer la performance d'une activité. Cette mesure peut être manuelle ou automatique, elle peut concerner un S.I. ou toute activité business. Elle permet au management de se rendre compte de l'évolution de l'élément mesuré (expl: temps de réponse d'une application, nombre de clients, nombre de campagne publicitaires par période...).

LDAP

Un LDAP (Lighweight Directory Access Protocol) permet généralement (en résumé) de stocker dans une base de donnée spécifique, toutes les informations concernant les individus travaillant pour une société ainsi que tous leurs droits d'accès.

MCS

Mobistar Corporate Solutions. Entité offrant des services orientés "grandes entreprises".

PKI

Public Key Infrastructure.

Problem form

Template standard utilisé pour décrire une erreur rencontrée dans une application et envoyé aux fournisseurs.

RATING

Nom d'une application qui mesure de degré de fiabilité et de solvabilité de nos clients, ceci pour accepter ou refuser une demande de service supplémentaire par le client (roaming mondial...)

RFP

Request for proposal

Roaming

Service qui permet à un utilisateur d'utiliser son GSM dans un autre pays que celui où il a contracté son abonnement.

SAX

Simple API for Xml. Mécanisme d'accès à un document XML. Ce mécanisme est sequentiel et basé sur des évènements.

Scheduler

Un scheduler est un outil de planification. Mobistar utilise *Unicenter work load* de la société Computer Associates. Un scheduler permet de lancer et de synchroniser le lancement de programmes *batch*.

SMS

Small Message System. Système permettant d'envoyer des messages court (170 caractères) de GSM à GSM.

SPRF

Standard Purchase Request Form: Un SPRF est un document standard de demande d'offre à un fournisseur.

système

Nom signifiant chez Mobistar: "serveur informatique" et également à l'origine du nom de l'équipe système (System Group: SYG). Cette équipe gére les serveurs Sun (UNIX), compaq (UNIX, NT), HP, UNISYS et VMS.

Team

Nom signifiant chez Mobistar: "équipe de Team member". Un team est sous la responsabilité hiérarchique d'un Team Leader. Nous utiliserons le mot "équipe" en lieu et place de "team" dans ce document.

Team leader

Nom signifiant chez Mobistar: "Responsable hiérarchique d'un team"

Team member

Nom signifiant chez Mobistar: "Employé de la société Mobistar"

Template

Un template définit la structure concernant un sujet. Exemple: Le template "DataBase" définira la structure des documents qui décriront les bases de données.

Unicenter work load

Application utilisée pour planifier, synchroniser et lancer les applications batch ainsi que pour récolter sur une console centrale tous les messages d'erreur importants de toutes les applications ainsi que de tous les systèmes, databases...

Utilisateur

Un utilisateur est une personne qui est autorisée à utiliser le S.I. Cette autorisation est subordonnée à une authentification. L'utilisateur est alors un utilisateur authentifié. Pour simplifier l'écriture, un utilisateur sera sous entendu authentifié.

W3C

World Wide Web Consortium (http://www.w3c.org)

Web

Voir WWW.

WAP

Wireless application protocol: Protocole léger utilisé par les GSM pour accéder aux applications dite "WAP". Une application WAP est une application pouvant fonctionner essentiellement sur un téléphone mobile mais pouvant être également accessible par l'internet et donc par tout équipement connecté à internet.

WILLY

Application de planning utilisée par IOP.

WWW (World Wide Web)

Littéralement, cela signifie "Toile d'araignée mondiale", la toile représentant le réseau internet. Le WWW est un réseau de documents HTML liés ensemble et se trouvant sur des serveurs localisés à travers le monde entier.

WYSIWYG

Application de planning utilisée par IOP.

XML

eXtensible Markup Language. Le XML est langage à base de balises permettant de définir de nouvelles balises pour identifier le texte et les données d'un document XML.

XSL

Langage permettant de décrire des règles de transformation applicable à un document XML.

Zone

Une zone est une liste qui peut être constituée de types. Ces types peuvent être: titre, texte, image, fichier, hyperlien interne, hyperlien externe).